



Article

# Soft Compression for Lossless Image Coding Based on Shape Recognition

Gangtao Xin <sup>1,2</sup>  and Pingyi Fan <sup>1,2,\*</sup> 

<sup>1</sup> Department of Electronic Engineering, Tsinghua University, Beijing 100084, China; xgt19@mails.tsinghua.edu.cn

<sup>2</sup> Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, China

\* Correspondence: fpy@tsinghua.edu.cn; Tel.: +86-010-6279-6973

**Abstract:** Soft compression is a lossless image compression method that is committed to eliminating coding redundancy and spatial redundancy simultaneously. To do so, it adopts shapes to encode an image. In this paper, we propose a compressible indicator function with regard to images, which gives a threshold of the average number of bits required to represent a location and can be used for illustrating the working principle. We investigate and analyze soft compression for binary image, gray image and multi-component image with specific algorithms and compressible indicator value. In terms of compression ratio, the soft compression algorithm outperforms the popular classical standards PNG and JPEG2000 in lossless image compression. It is expected that the bandwidth and storage space needed when transmitting and storing the same kind of images (such as medical images) can be greatly reduced with applying soft compression.

**Keywords:** lossless image compression; information theory; statistical distributions; compressible indicator function; image set compression



**Citation:** Xin, G.; Fan, P. Soft Compression for Lossless Image Coding Based on Shape Recognition. *Entropy* **2021**, *23*, 1680. <https://doi.org/10.3390/e23121680>

Academic Editors: Amelia Carolina Sparavigna and Armando J. Pinho

Received: 31 October 2021  
Accepted: 10 December 2021  
Published: 14 December 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Image compression is to reduce the required number of bits as much as possible when representing an image. In this process, the fidelity of the reconstructed image and original image should be higher than the reference value. Image compression often consists of two parts, encoding and decoding. Encoding is to convert the input image into a binary code stream with a coding method, while decoding, the reverse process of encoding, aims to restore the original image from the binary code stream.

There are two categories of image compression: lossy compression and lossless compression. Lossy compression allows the reconstructed image to be different from the original image, but it is still visually similar. However, lossless compression requires the reconstructed image to be exactly the same as the original image, which leads to the compression ratio being much smaller than that of lossy compression. Although the compression ratio of lossy compression is higher, lossless compression is significant in many fields. Lossless compression should be applied when errors cannot be tolerated or the image has significant value, such as medical images, precious cultural relics, deep space exploration, deep sea exploration and digital libraries.

Supposing that we regard an image as a random process, the minimum expected codeword length per pixel  $L_n^*$  will approach the entropy rate asymptotically, as shown in Formula (1). However, for an actual image, the upper and lower bounds of Formula (1) cannot be theoretically calculated because one cannot know the spatial correlation of pixels clearly.

$$\frac{H(X_1, X_2, \dots, X_n)}{n} \leq L_n^* < \frac{H(X_1, X_2, \dots, X_n)}{n} + \frac{1}{n} \quad (1)$$

where  $H(X_1, X_2, \dots, X_n)$  is the joint entropy of the symbol series  $\{X_i, i = 1, 2, \dots, n\}$ .

It is impossible to reach the entropy rate for an image. All we can do is to make great efforts to get close to it. Xin et al. [1] proposed soft compression based on information theory and statistical distribution. It uses shapes and locations to reflect the spatial correlation of an image, trying to achieve better compression performance.

In the literature, most of the image compression methods mainly consider three aspects to reduce the required number of bits when representing an image: coding redundancy, spatial redundancy and irrelevant information. Coding redundancy refers to the diverse probability of each pixel value in an image so that the average length can be reduced from the perspective of coding. Spatial redundancy means that pixels are spatially related. The repeated information can be omitted because the pixel is similar to or depends on adjacent pixels [2]. Irrelevant information refers to an image containing information irrelevant to the human visual system or purpose, which leads to redundancy. Image compression techniques usually improve the algorithm performance from one or several aspects.

### 1.1. Image Compression Method

Huffman coding [3] is an extraordinary method to eliminate coding redundancy for a stream of data. Arithmetic coding [4] and Golomb coding [5] are also approaches to eliminating coding redundancy. They all require accurate probability models of input symbols. Run-length coding [6] represents runs of identical intensities by a new coding value and length, but it may result in data expansion when there are few runs of identical intensities [2]. LZW coding [7] is a method to remove spatial redundancy, assigning fixed-length codewords to variable length sequences of source symbols, but it is easy to cause data explosion, especially when the input is of a large size or irregular.

Image predictive coding is a means of transforming spatial redundancy into coding redundancy through prediction error, which is an entry point of image compression. The paper [8] applies prediction to discrete wavelet transform subbands. In [9], it predicts the probability of a high-resolution image, conditioned on the low-resolution input, and uses entropy coding to compress the super-resolution operator. The Consultative Committee for Space Data Systems (CCSDS) [10] is a multi-national forum for the development of communications and data systems standards for spaceflight, which has proposed several excellent image lossless compression algorithms [11,12].

Transform coding [13–15] maps an image from the spatial domain to transform domain, and then encodes the coefficients of the transform domain to achieve the compression effect. It reduces the irrelevant information in an image from the visual point of view. As a tool of multi-resolution analysis, wavelet coding [16–18] has been widely concerned and applied. In [19], it uses both wavelet and fractional transforms for lossless image compression. In [20], it designs a reversible integer-to-integer wavelet filter to achieve the effect of lossless compression. In [21], it describes edge-based and prediction-based transformations for image compression.

With the development of neural networks, image compression methods based on learning have received a lot of attention [15,22–24]. Recent works are mainly in the area of lossy compression, which are based on convolutional neural networks (CNNs) [25–29], recurrent neural networks (RNNs) [30], generative adversarial networks (GANs) [31] and the context model [32–34]. Learning-based lossless image compression methods [35–39] use neural networks instead of the traditional encoder and decoder to achieve image compression. PixelCNN [40] and PixelCNN++ [41] as well as the methods based on bits-back coding [42,43] and flow models [44,45] shorten the distance between information theory and machine learning. In [46], it proposes a fully parallel hierarchical probabilistic model with auxiliary feature representations. The neural network of long short-term memory (LSTM) can also be used to build a predictor for lossless image compression [47].

As for image coding standards, there are some mature instances (PNG [48], JPEG XR [49], JPEG-LS [50], and WebP [51]). JPEG [52] and JPEG2000 [53] are based on discrete cosine transform [54] and wavelet transform [55], respectively. FLIF [56] is based on meta-adaptive near-zero integer arithmetic coding.

### 1.2. Related Work

Soft compression has two special properties. (1) It uses shapes and corresponding locations to represent an image. (2) Its codebook is generated through data-driven means. The earliest coding approaches with symbols and locations can be traced back to symbol-based coding [57]. A picture is denoted as a set of frequently occurring sub-images, called symbols. Storing repeated symbols only once can compress images significantly, especially in document storage, where the symbols are usually character bitmaps that are repeated many times. However, symbol-based coding is hard to generalize to other scenarios, owing to the need of redesigning symbols. Some methods are also based on shape coding [58,59], but none of them consider both shapes and locations at the same time. Fractal block coding [60] relies on the assumption that image redundancy can be efficiently exploited through self-transformability on a blockwise basis and it can approximate an original image by a fractal image. However, it is mainly used in lossy compression because it is tough to find a great deal of identical blocks from only one image.

Finding similar features from a database has been an active research topic in the field of image compression in recent years. In [61], an off-the-shelf image database is used to find patches that are visually similar to each region of interest of the unknown input image, according to associated local descriptors. These patches are then warped into the input image domain according to interest region geometry and seamlessly stitched together. In [62], the authors make use of external image contents to reduce visual redundancy among images with SIFT descriptors [63]. In [64], a method is proposed for cloud-based image coding that no longer compresses images pixel by pixel and instead tries to describe images and reconstruct them from a large-scale image database via the descriptions. In [65], the authors adopt a semi-local approach to exploit inter-image correlation by using information from external image similarities. In [66], a cloud storage system is proposed that reduces the storage cost of all uploaded JPEG photos at the expense of a controlled increase in computation mainly during the download of a requested image subset. In [67], it proposes a novel framework for image set compression based on the rate-distortion optimized sparse coding.

### 1.3. Soft Compression

Soft compression was first proposed in [1], using shapes and locations to represent a binary image. The set of shapes used in soft compression is not designed by experts, but searched from a dataset. Different datasets may have diverse codebooks, which ensures the adaptability of soft compression. Moreover, the codebook corresponding to each dataset is complete, containing all the possibilities of the smallest shape, which makes soft compression workable. Due to the adaptability and completeness of codebooks of soft compression, they can always achieve lossless compression for any image and any codebook. When the codebook and image match well, it will result in a better compression ratio.

The main idea of soft compression is to represent an image with a set of triplets  $(x_i, y_i, S_i)$ , where  $(x_i, y_i)$  denotes the position of shape  $S_i$  in an image. The set of shapes is obtained by searching in the training set. After that, the set of codewords and the codebook can be obtained by variable length coding for the set of shapes according to the size and frequency of each shape. When an image is encoded, it is transformed into a set of triplets  $(x_i, y_i, C_i)$  according to the codebook, where  $C_i$  is the codeword of shape  $S_i$ . On the other hand, we also require to decode the compressed data into a set of triplets  $(x_i, y_i, S_i)$  according to the codebook when decoding. Finally, we fill shapes in the corresponding locations to reconstruct the original image.

Soft compression is instrumental in reducing storage space and communication bandwidth in the process of transmitting and storing the same kind of images. When two sides communicate, the transmitter only needs to send the compressed data instead of the whole picture to the receiver in the case that both sides have identical codebooks.

In this paper, we try to answer the following two fundamental problems for lossless image compression and design a novel image coding algorithm based on soft compression, outperforming the popular classical standards, PNG and JPEG2000.

① How do we detect an image to be compressible in theory? In other words, what is the value of the compressible indicator function for an image?

② If one image is compressible, how do we find a way to compress it through increasing the value of the compressible indicator function?

This paper is organized as follows. We first introduce a new concept, a compressible indicator function with regard to images based on information theory. Then, we use it to evaluate the performance of soft compression in Section 2. In Section 3, some soft compression algorithms for binary image, gray image and multi-component image are proposed. Then, we give the experimental results and theoretical analysis in Section 4. Finally, we conclude this paper in Section 5.

## 2. Theory

Digital images have coding and spatial redundancy, which makes compression feasible. Soft compression is committed to eliminating these two kinds of redundancy simultaneously by filling an image with shapes. In this section, we introduce the theory of soft compression.

### 2.1. Preliminary

#### 2.1.1. Information Theory

Information theory provides the answer to the lower bound of data compression. For an image, the minimum number of bits required per pixel is given by formula (1), which is the entropy rate of a random process.

**Definition 1.** Let  $Z$  be a discrete random variable with alphabet  $\mathcal{Z}$  and probability mass function  $p(z) = \Pr\{Z = z\}, z \in \mathcal{Z}$ . The entropy [68]  $H(Z)$  of a discrete random variable  $Z$  is defined as

$$H(Z) = - \sum_{z \in \mathcal{Z}} p(z) \log p(z) \quad (2)$$

Entropy is a measure of the average uncertainty of a random variable. Moreover, it points out the minimum cost of encoding the random variable [69]. In this paper, we take all logarithms to base 2 so that entropy is measured in bits unless otherwise specified.

**Definition 2.** Suppose that  $Z$  is a random variable with only two events, i.e.,

$$Z = \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } 1 - p \end{cases} \quad (3)$$

Then the entropy of  $Z$  is given by

$$H(Z) = -p \log p - (1 - p) \log(1 - p) \stackrel{\text{def}}{=} H(p) \quad (4)$$

Note that  $H(p)$  is a concave function of  $p$  and equals 1 when  $p = 0.5$ . In the case of  $p = 0$  or  $p = 1$ ,  $H(p)$  reaches its minimum value of 0. Moreover, the random variable becomes a constant due to the lack of randomness.

If each pixel in an image is independently and identically distributed, the minimum expected number of bits required is the entropy. However, for an actual image, the probability distribution of each pixel cannot be independently and identically distributed. Due to the spatial correlation, the minimum expected number of bits required for a pixel is the entropy rate of the random process corresponding to an image. How to evaluate it is still an open problem in the literature.

### 2.1.2. Image Fundamentals

Let  $I$  denote a digital image with intensity levels in the range  $[0, D - 1]$  whose row and column dimensions are  $M$  and  $N$ , respectively.  $r_k$  is the  $k$ -th intensity value.  $n_k$  is the number of pixels with intensity  $r_k$  in the image  $I$  [2]. We define  $X$  as a discrete random variable with probability mass function  $p(x_k)$

$$p(x_k) = \Pr\{X = r_k\} = \frac{n_k}{MN} \quad k = 0, 1, 2, \dots, D - 1 \quad (5)$$

$X$  reflects the frequency distribution of pixel intensity values in an image.

We define  $Y$  as the same random variable as  $X$ , but  $Y$  removes event  $r_0$ . Let  $p = \Pr\{X = r_0\}$ , then

$$p(y_k) = \frac{p(x_k)}{1 - p} \quad k = 1, 2, \dots, D - 1 \quad (6)$$

$Y$  indicates the frequency distribution of remaining intensity values with removing  $r_0$  from  $X$ . When  $p = 1$ ,  $X$  will change from a random variable to a constant. For this reason, we mainly consider the case where  $p < 1$ .

**Lemma 1.** Let  $H(X)$  and  $H(Y)$  denote the entropy of  $X$  and  $Y$ , respectively. Then,

$$H(Y) = \frac{H(X) - H(p)}{1 - p} \quad (7)$$

where  $H(p)$  comes from Definition 2.

**Proof.**

$$H(Y) = - \sum_{k=1}^{D-1} p(y_k) \log p(y_k) \quad (8)$$

$$= - \sum_{k=1}^{D-1} p(y_k) \log p(x_k) + \sum_{k=1}^{D-1} p(y_k) \log(1 - p) \quad (9)$$

$$= - \sum_{k=1}^{D-1} \frac{p(x_k)}{1 - p} \log p(x_k) + \log(1 - p) \quad (10)$$

$$= \frac{1}{1 - p} \left[ - \sum_{k=1}^{D-1} p(x_k) \log p(x_k) \right] + \log(1 - p) \quad (11)$$

$$= \frac{1}{1 - p} [H(X) + p \log p] + \log(1 - p) \quad (12)$$

$$= \frac{H(X) + p \log p + (1 - p) \log(1 - p)}{1 - p} \quad (13)$$

$$= \frac{H(X) - H(p)}{1 - p} \quad (14)$$

□

### 2.2. Soft Compression

Soft compression is a lossless image compression method which aims to fill images with shapes and locations. The purpose of soft compression is to find essential and representative shapes.

We define  $\mathcal{S}$  and  $\mathcal{C}$  as the finite set of shapes and codewords with soft compression, respectively. Let  $T$  denote the total number of operations in the process of filling an image  $I$  [1]. Let  $S_i \in \mathcal{S}, i = 1, \dots, T$  be the shape used in the  $i$ -th operation.  $C_i \in \mathcal{C}, i = 1, \dots, T$  is the codeword of  $S_i$ . The location of shape  $S_i$  is defined as the pixel coordinate pair  $(x_i, y_i)$ .

We use  $F_i(S_i), i = 1, \dots, T$  to represent filling an image with shape  $S_i$  at position  $(x_i, y_i)$  in the  $i$ -th operation.

Then, soft compression can be formulated as the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^T [l(C_i) + l(x_i, y_i)] \\ \text{s.t.} \quad & I = \sum_{i=1}^T F_i(S_i) \end{aligned} \quad (15)$$

where  $l(C_i)$  is the number of bits corresponding to  $C_i$ , and  $l(x_i, y_i)$  is the length of bits needed to represent  $(x_i, y_i)$ . The constraint reflects that the original image  $I$  can be reconstructed with filling shapes through  $T$  operations. That is, soft compression is lossless. The core goal of designing soft compression algorithm is to find  $\mathcal{S}$  and  $\mathcal{C}$  so as to encode images efficiently and effectively.

**Definition 3.** We define the compressible indicator function (CIF) with respect to  $p$  as

$$C(p) = \sup_k \frac{H(p(x_k))}{1 - p(x_k)} \quad k = 0, 1, 2, \dots, D - 1 \quad (16)$$

Without loss of generality, we assume  $p = p(x_0) = \sup_k p(x_k)$ . Then,

$$C(p) = \frac{H(p)}{1 - p} \quad p \in (0, 1) \quad (17)$$

Moreover, we define compressible indicator value (CIV) as the value of CIF.

Compressible indicator function  $C(p)$  is derived from Lemma 1, which indicates the sparsity of an image. The larger it is, the more capacity can be compressed. The basic properties of the compressible indicator function can be summarized as follows.

**Theorem 1.** ①  $C(p) > 0$

②  $C(p)$  is monotonically increasing.

**Proof.** ①  $0 < p < 1$  implies that  $H(p) > 0$ , which can reach the conclusion that  $C(p) > 0$ .

② The derivative of  $C(p)$  with respect to  $p$  is

$$C'(p) = -\frac{\log p}{(1 - p)^2} > 0 \quad (18)$$

and therefore,  $C(p)$  is monotonically increasing.  $\square$

**Definition 4.** We use  $B_{nb}$  to denote the number of bits required to encode an image with natural binary code,  $B_{hf}$  for Huffman coding. Similarly,  $B_{sc}^n$  is for soft compression where the size of the shapes ranges from 1 to  $n$ , i.e., soft compression with  $n$  order.  $B_{hf, \min}$  and  $B_{sc, \min}^n$  represent the minimum values of  $B_{hf}$  and  $B_{sc}^n$ , respectively.

Let  $L_{hf}$  be the average number of bits required to represent each pixel with Huffman coding.  $L_{sc}^1$  is for soft compression where the size of all shapes is one. Then,

$$B_{nb} = MN \log D \quad (19)$$

$$B_{hf} = MN L_{hf} \quad (20)$$

$$B_{sc}^1 = MN(1-p)(L_{sc}^1 + L_W) \quad (21)$$

$$B_{hf,min} = MNH(X) \quad (22)$$

$$B_{sc,min}^1 = MN(1-p)(H(Y) + L_W) \quad (23)$$

where  $L_W$  is the average number of bits required to represent a location with soft compression.

**Theorem 2.** If  $C(p) \geq L_W$  and  $H(X) > 0$ , the minimum number of bits needed with 1 order soft compression is less than that with Huffman coding, i.e.,  $B_{sc,min}^1 \leq B_{hf,min}$ . The relative compression ratio  $R'$  is

$$R' = 1 + (1-p) \frac{C(p) - L_W}{H(X)} \quad (24)$$

**Proof.**

$$B_{sc,min}^1 = MN(1-p)(H(Y) + L_W) \quad (25)$$

$$= MN(1-p) \left[ \frac{H(X) - H(p)}{1-p} + L_W \right] \quad (26)$$

$$= MN[H(X) - H(p) + (1-p)L_W] \quad (27)$$

$$= B_{hf,min} + MN[(1-p)L_W - H(p)] \quad (28)$$

Equation (26) uses Lemma 1.

To obtain the result  $B_{sc,min}^1 \leq B_{hf,min}$ , we can reach the conclusion that

$$\frac{H(p)}{1-p} = C(p) \geq L_W \quad (29)$$

From Theorem 1, we know that  $C(p)$  increases monotonically in  $(0,1)$ . Due to the non-negativity of entropy and the trivial case for  $H(X) = 0$ , we mainly consider the case where  $H(X) > 0$ , then

$$R' = \frac{B_{hf,min}}{B_{sc,min}^1} \quad (30)$$

$$= \frac{B_{sc,min}^1 - MN[(1-p)L_W - H(p)]}{B_{sc,min}^1} \quad (31)$$

$$= 1 - \frac{MN[(1-p)L_W - H(p)]}{B_{sc,min}^1} \quad (32)$$

$$= 1 - \frac{MN[(1-p)L_W - H(p)]}{MNH(X)} \quad (33)$$

$$= 1 + \frac{H(p) - (1-p)L_W}{H(X)} \quad (34)$$

$$= 1 + (1-p) \frac{C(p) - L_W}{H(X)} \quad (35)$$

It completes the proof.  $\square$

Theorem 2 provides the threshold relationship between  $L_W$  and  $C(p)$ . When  $L_W$  is less than this threshold, the minimum number of bits needed to represent an image with soft compression is lower than that of Huffman coding. In general, we use the minimum value to approximately replace the actual value needed for compression, which is convenient

for theoretical analysis. Theorem 2 indicates that for an image whose  $C(p) \geq L_W$ , soft compression is better than Huffman coding in terms of the compression ratio. It also points out that the higher the compressible indicator value, the higher the compression ratio.

Figure 1 illustrates the relationship between compressible indicator function  $C(p)$  and  $p$ . Theorem 2 points out the suitability of applying soft compression. Given an image, we should first evaluate the compressible indicator value according to  $p$ , and then judge whether it is suitable for soft compression. That is to say, if  $(p, L_W)$  is in the gray area, the minimum number of bits required with 1 order soft compression is less than that with Huffman coding. As shown in Figure 1, it answers the first fundamental problem of lossless image compression.

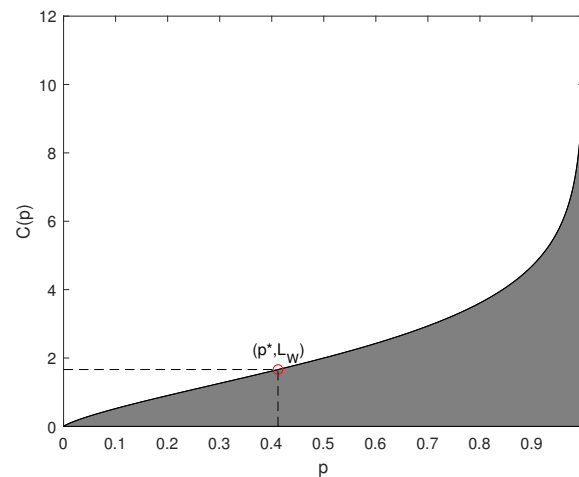


Figure 1. Compressible indicator function versus  $p$ .

**Lemma 2.** Let  $Y_n$  represent the frequency distribution of a shape whose size is  $n$  in an image, then

$$H(Y_n) \leq nH(Y) \tag{36}$$

**Proof.** From the independence bound on entropy, one can come to this conclusion. □

**Theorem 3.** If  $L_W$  is a constant for different orders of soft compression, then  $B_{sc,min}^n \leq B_{sc,min}^1$ .

**Proof.** We use  $N_1$  to represent the number of shapes with size 1, and  $N_n$  is the number of shapes with size  $n$ . The derivation can be seen from (37) to (41). □

$$B_{sc,min}^n = N_1[H(Y) + L_W] + N_2[H(Y_2) + L_W] + \dots + N_n[H(Y_n) + L_W] \tag{37}$$

$$\leq [N_1 + 2N_2 + \dots + nN_n]H(Y) + [N_1 + N_2 + \dots + N_n]L_W \tag{38}$$

$$\leq [N_1 + 2N_2 + \dots + nN_n][H(Y) + L_W] \tag{39}$$

$$= MN(1 - p)[H(Y) + L_W] \tag{40}$$

$$= B_{sc,min}^1 \tag{41}$$

Theorems 2 and 3 inspire us that in image compression, we can improve the compression ratio from two aspects: one is to increase the compressible indicator value of an image, and the other is to reduce the number of bits required to represent locations. On the one hand, the compressible indicator value can be increased by predictive coding. On the other hand, Golomb coding may be a promising method for encoding the distance between each location and the previous location.

### 3. Implementation Algorithm

In this section, we try to answer the second problem mentioned in Section 1, finding some efficient ways to improve the compression ratio with raising the compressible in-



indicator value. We introduce the soft compression algorithm for different image formats, including binary image, gray image and multi-component image. In fact, three different algorithms vary in specific steps but all try to fill images with shapes and corresponding locations.

### 3.1. Binary Image

The binary image is quite suitable for encoding with soft compression because each pixel has only two intensity values. The probability of  $r_0$  can always be greater than or equal to 0.5 (through reverse operation), which ensures that the compression indicator value is greater than or equal to 2.

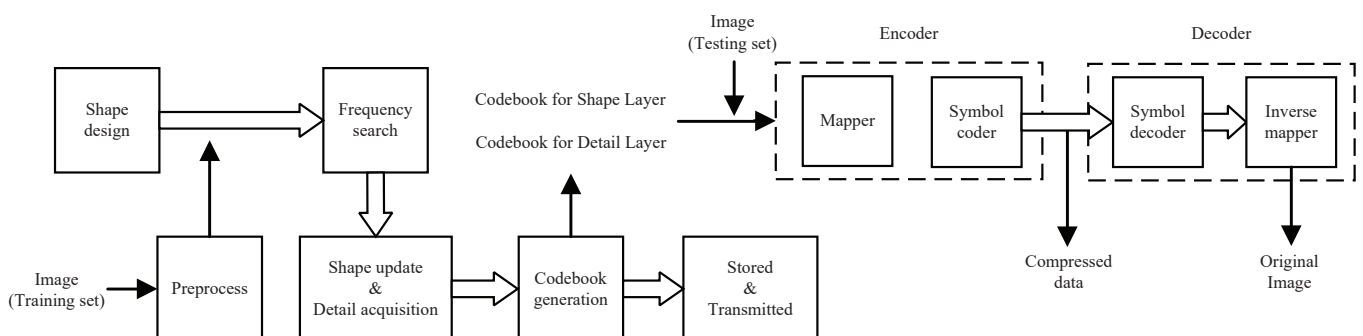
The soft compression algorithm for a binary image was proposed in [1], which had excellent compression performance. Although [1] introduced the algorithm, they did not provide the theoretical analysis. We analyze the experimental results with compressible indicator function in Section 4.

### 3.2. Gray Image

One of the most important steps of the soft compression algorithm for a gray image is to divide the images into two layers, the shape layer and detail layer. In fact, the compressible indicator value of the shape layer is usually higher, so the combinations of shapes and locations are used to encode. Meanwhile, the compressible indicator value of the detail layer is relatively lower, and other common coding methods can be adopted for encoding. The soft compression algorithm for a gray image consists of predictive coding, negative-to-positive mapping, layer separation, shape search, codebook generation and so on. We first introduce the overall architecture, followed by the vital steps.

#### 3.2.1. Overall Architecture

For soft compression algorithms, the codebook is very important. It directly determines the efficiency and performance of image compression. The algorithm consists of two parts, training and testing. The purpose of the training phase is to generate the codebook. In the testing phase, codebooks are used to encode and decode images to evaluate the performance of the algorithm. Figure 2 summarizes the overall architecture of the soft compression algorithm for a gray image. We design the set of shapes firstly, and then update the frequency of each candidate shape in the training set. After obtaining the final set of shapes and corresponding frequency, we generate the codebook for the shape layer. In training, we also acquire the codebook for the detail layer at the same time. Codebooks are used in the encoder and decoder, which are stored or transmitted for subsequent usage.



**Figure 2.** The overall procedure of soft compression algorithm for gray image including training and testing. The shape design, frequency search, shape update, detail acquisition, codebook generation, and saved codebook are the training stage. On the other hand, the encoder and decoder are the testing stage.

In testing, the image is compressed through the encoder. When the sender wants to communicate with the receiver, it firstly transmits the codebooks. After both sides of the communication have the same codebook, the transmitted content can be the compressed

data instead of the original image. The receiver receives the compressed data after storage or transmission. Due to the completeness of codebooks, the recovered image is exactly the same as the original image, which ensures lossless compression.

### 3.2.2. Predictive Coding and Negative-to-Positive Mapping

Predictive coding is an efficient way to transform spatial redundancy into coding redundancy with prediction. The core idea of predictive coding is to calculate the prediction value according to the spatial correlation, which aims to encode the prediction error. Negative-to-positive mapping maps the prediction error from negative to positive for layer separation.

Let  $I(x, y)$ ,  $I_p(x, y)$  and  $I_E(x, y)$  be the pixel intensity value, predictive value, prediction error at position  $(x, y)$  in an image  $I$ , respectively. We utilize the gradient adjusted prediction method [70] based on the gradient of the intensity function, which is shown in Formulas (42) and (43) (for simplicity, we use  $I_{x,y}$  for  $I(x, y)$ ). Figure 3 illustrates the spatial correlation between pixels.

		$I_{(x-2,y)}$	$I_{(x-2,y+1)}$
	$I_{(x-1,y-1)}$	$I_{(x-1,y)}$	$I_{(x-1,y+1)}$
$I_{(x,y-2)}$	$I_{(x,y-1)}$	$I_p(x,y)$	

Figure 3. The spatial correlation between pixels with predictive coding.

After obtaining the predictive value, one can calculate the prediction error with Formula (44). The range of the prediction error is  $[-D + 1, D - 1]$ , which is different from the pixel intensity value  $[0, D - 1]$ . We use Formula (45) to map the prediction error from negative to positive, which is conducive to the subsequent procedures.

$$\begin{aligned}
 d_h &= |I_{x,y-1} - I_{x,y-2}| + |I_{x-1,y} - I_{x-1,y-1}| + |I_{x-1,y} - I_{x-1,y+1}| \\
 d_v &= |I_{x,y-1} - I_{x-1,y-1}| + |I_{x-1,y} - I_{x-2,y}| + |I_{x-1,y+1} - I_{x-2,y+1}| \\
 d_s &= d_v - d_h
 \end{aligned}
 \tag{42}$$

$$I_p(x, y) = \begin{cases} I_{x-1,y} & \text{if } d_s < -80 \\ \frac{1}{4}I_{x,y-1} + \frac{3}{4}I_{x-1,y} + \frac{1}{8}I_{x-1,y+1} - \frac{1}{8}I_{x-1,y-1} & \text{if } -80 \leq d_s < -32 \\ \frac{3}{8}I_{x,y-1} + \frac{5}{8}I_{x-1,y} + \frac{3}{16}I_{x-1,y+1} - \frac{3}{16}I_{x-1,y-1} & \text{if } -32 \leq d_s < -8 \\ \frac{1}{2}I_{x,y-1} + \frac{1}{2}I_{x-1,y} + \frac{1}{4}I_{x-1,y+1} - \frac{1}{4}I_{x-1,y-1} & \text{if } -8 \leq d_s \leq 8 \\ \frac{5}{8}I_{x,y-1} + \frac{3}{8}I_{x-1,y} + \frac{3}{16}I_{x-1,y+1} - \frac{3}{16}I_{x-1,y-1} & \text{if } 8 < d_s \leq 32 \\ \frac{3}{4}I_{x,y-1} + \frac{1}{4}I_{x-1,y} + \frac{1}{8}I_{x-1,y+1} - \frac{1}{8}I_{x-1,y-1} & \text{if } 32 < d_s \leq 80 \\ I_{x,y-1} & \text{if } d_s > 80 \end{cases}
 \tag{43}$$

$$I_E(x, y) = I(x, y) - I_p(x, y)
 \tag{44}$$

$$I'(x, y) = \begin{cases} 2I_E(x, y) & I_E(x, y) \geq 0 \\ -2I_E(x, y) - 1 & I_E(x, y) < 0 \end{cases}
 \tag{45}$$

The proportion of  $r_0$  increases with predictive coding and mapping, which also leads to the increase in the compressible indicator value. The images will be more conducive to encoding with the soft compression algorithm. Another reason for adopting predictive coding and mapping is that the same image will have the same result after these two operations. The reversibility of each step ensures the losslessness of the soft compression algorithm.

### 3.2.3. Layer Separation

Soft compression is suitable for images with a large compressible indicator value. After the previous steps, the compressible indicator value of an image is greatly improved. We continue to improve it. On the one hand, we observe that the probability of the prediction error decreases as the value increases. On the other hand, through a proper separation, one part with a higher compressible indicator value and another with a lower value can be generated.

Layer separation and bit-plane coding [71] are similar, but there are essential differences. Bit-plane coding focuses on decomposing a multilevel image into a series of binary images and compressing each binary image via one of several well-known binary compression methods. Bit-plane coding produces many layers. However, layer separation produces only two layers, shape layer  $I'_S$  and detail layer  $I'_D$ .

$I'$  is divided into shape layer  $I'_S$  and detail layer  $I'_D$  via Formulas (46) and (47).

$$I'_S(x, y) = I'(x, y) // 2^l \quad (46)$$

$$I'_D(x, y) = I'(x, y) \% 2^l \quad (47)$$

where  $//$  and  $\%$  represent the quotient and remainder operation, respectively. Layer interface  $l$  is a constant between 0 and  $\log D$ , which can be given in advance with searching or experience.

Because the compressible indicator value of the shape layer is usually higher, the combinations of locations and shapes are used for encoding. The compressible indicator value of the detail layer is relatively lower, and other coding methods, such as block coding and arithmetic coding, can be used for compression.

### 3.2.4. Shape Search and Codebook Generation


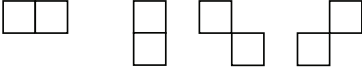
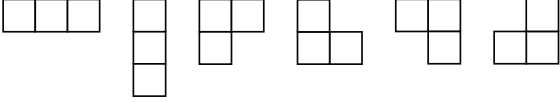
The set of shapes with the soft compression algorithm directly determines the compression performance and coding efficiency of images. How to find the shape is vital.  $A$  is an  $m \times n$  matrix whose components are in  $[0, 2D - 2]$ .  $u_i$  and  $v_j$  are vectors, representing the  $i$  row and  $j$  column of  $A$ , respectively. The matrix whose  $u_i$  and  $v_j$  follow (48) and (49) is suitable for designing shapes.

$$\|u_i\|_0 \geq \frac{n}{2} \quad \forall 1 \leq i \leq m \quad (48)$$

$$\|v_j\|_0 \geq \frac{m}{2} \quad \forall 1 \leq j \leq n \quad (49)$$

Formulas (48) and (49) indicate that the number of non-zero elements in all rows and columns must be no less than half of the row size and column size, respectively. By keeping only the non-zero value blocks in a matrix and removing the zero value blocks, one can obtain the candidate shape. This method can avoid the situation that different matrices generate the same shape.

As illustrated in Figure 4, there is a part of shapes. These shapes are classified by size without considering intensity values. Combining them with the error intensity value  $[1, 2D - 2]$  can generate the shapes in actual use.

Number of pixels	Shapes without considering intensity values
1	
2	
3	
...	...

**Figure 4.** Shapes generated without considering intensity values according to the criteria (classified by the number of pixels). These are not all shapes in the set but only a part of it.

However, not all shapes appear in the final shape set. In fact, only a small number of shapes can be retained all the time. In training, the set of shapes is updated dynamically. The shape with less frequency is deleted in order to ensure that the size of shape set is controllable.

After obtaining the set of shapes, the codebook for the shape layer can be generated according to the frequency and size of each shape. While searching for the shape set, we also count the frequency distribution of the pixel intensity value in the detail layer so as to generate the codebook for the detail layer. The process of the training phase is visually shown in Algorithm 1, which connects the steps in Sections 3.2.2–3.2.4. When both sides of communication have the same codebook prior to the message exchanges, the transmitter can directly send compressed data instead of the whole image to the receiver, which is able to reduce the communication quantity and storage space. That is, it can save the communication bandwidth while keeping the same message change rate.

---

**Algorithm 1** The training part of the soft compression algorithm for gray image.

---

**Input:**  $W$  images with size  $M \times N$

**Output:** The codebook for the shape layer and detail layer

Preprocess: predictive coding, negative-to-positive mapping and layer separation

**for**  $Z \leftarrow 1$  **to**  $W$  **do do**

**for** matrix size  $(u, v) \leftarrow 1 \times 1$  **to**  $m \times n$ , image coordinate  $(i, j) \leftarrow 1 \times 1$  **to**  $M \times N$  **do**

**if**  $I'_S[i, j : i + u, j + v]$  satisfies (48) **and** (49) **then**

Get the shape  $S := I'_S[i, j : i + u, j + v]$

**if**  $S$  in the codebook **then**

Update the frequency of  $S$  in the codebook

**else**

Add  $S$  to the codebook

**end if**

**end if**

**end for**

Remove low-weight shapes based on frequency and size

Count the distribution of pixel values in the detail layer  $I'_D$

**end for**

Generate the codebooks

---

### 3.2.5. Golomb Coding for Locations

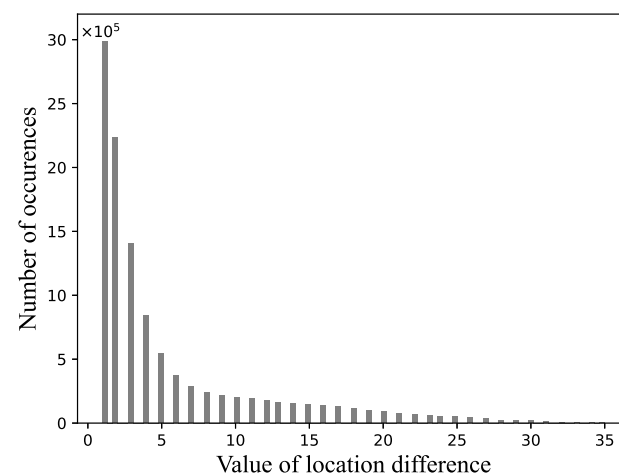
A set of locations are generated when the shape layer is encoded. We use Golomb coding to encode the distance difference between each location and the previous location. Golomb coding [5] was designed for non-negative integer input with geometric probability distribution. We use it in the following steps.

- **Step 1.** Calculate the distance difference  $\Delta$  from the previous location.
- **Step 2.** Get a positive integer  $m$  by giving or searching in advance.
- **Step 3.** Form the unary code of quotient  $\lfloor \Delta/m \rfloor$ . (The unary code of an integer  $q$  is defined as  $q$  1s followed by a 0.)
- **Step 4.** Let  $k = \lceil \log_2 m \rceil, c = 2^k - m, r = \Delta \bmod m$ , and compute truncated remainder  $r'$  such that

$$r' = \begin{cases} r \text{ truncated to } k - 1 \text{ bits} & 0 \leq r < c \\ r + c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases} \quad (50)$$

- **Step 5.** Concatenate the results of steps 3 and 4.

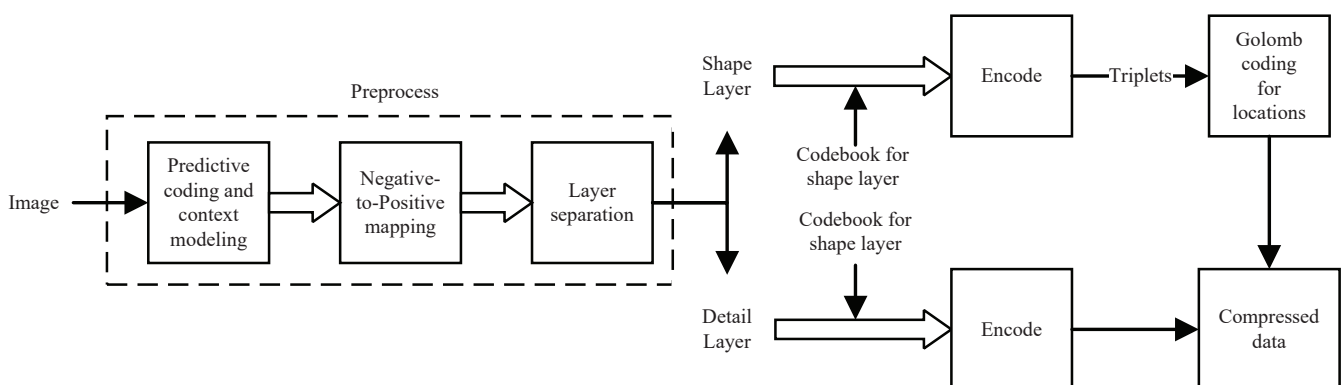
The location difference obtained with the soft compression algorithm approximately obeys geometric probability distribution. Figure 5 shows the empirical frequency distribution on Fashion-MNIST dataset with the soft compression algorithm for gray image. Under the prior information, Golomb coding reduces the space for storing location differences.



**Figure 5.** The frequency distribution of location difference on Fashion-MNIST dataset with soft compression algorithm for gray image.

### 3.2.6. Encoder and Decoder

Encoder starts with preprocessing, which consists of the steps mentioned before (predictive coding, negative-to-positive mapping and layer separation). Secondly, two codebooks are used to encode the shape layer and detail layer, respectively. Thirdly, locations of the shape layer are encoded with Golomb coding. Finally, the compressed data can be obtained by connecting the coding results of the two layers. Figure 6 shows the entire process of the encoder.



**Figure 6.** The procedure of encoder. It uses two different codebooks to encode the shape layer and the detail layer, respectively.

Figure 7 illustrates the composition of the compressed data. The header part contains information about the height and width of an image, as well as the layer interface. The shape layer data and detail layer data carry the encoding results of these two layers, respectively.

The decoder adopts the opposite structure to the encoder. The original image can be reconstructed through decoding. Firstly, the shape  $S_i$  is obtained according to the codebook for the shape layer, and then the shape  $S_i$  is filled in the location  $(x_i, y_i)$ . The shape layer  $I'_S$  is acquired by repeating  $T$  operations. Secondly, the detail layer  $I'_D$  is decoded from the compressed data with the codebook for the detail layer. Finally, the original image can be reconstructed by merging  $I'_S$  and  $I'_D$ , positive-to-negative mapping and anti-predictive coding.

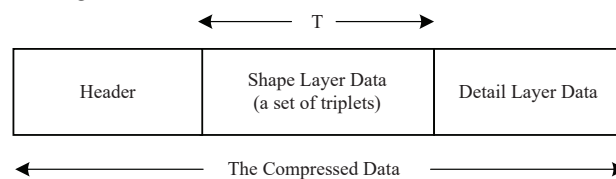


Figure 7. The composition of compressed data.

### 3.2.7. Concrete Example

In order to describe the algorithm more intuitively, we use a specific example to illustrate the encoding process of the shape layer, in other words, how it changes from a digital image to the binary data. Figure 8 illustrates the process of filling an image with shapes in the codebook. The image on the left is an  $8 \times 8$  shape layer. It is divided into several shapes. Each shape is marked with a different color, as shown on the right. We use the coordinate of the pixel in the upper left corner to represent the location of the whole shape. Therefore, the locations of these shapes are  $\{(x_i, y_i) : (0, 0), (0, 6), (3, 5), (4, 2), (5, 4), (6, 0), (6, 5)\}$ . Using relative values to represent locations can reduce coding costs. In this case, the locations can be re-expressed as  $\{(x_i, y_i) : (0, 0), 6, 23, 5, 10, 4, 5\}$ . Except for the first location, the rest are represented by relative values (difference from the previous location).

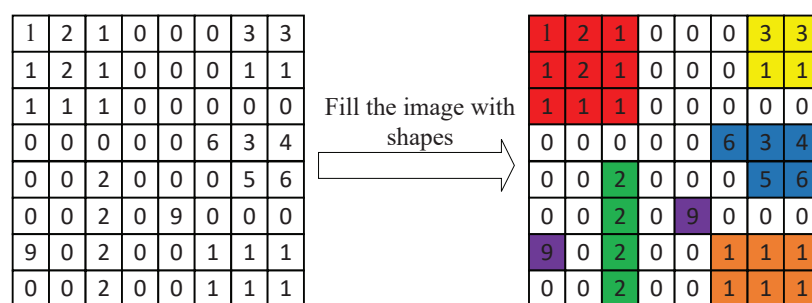


Figure 8. An example of the split of the shape layer. The image on the left is the original  $8 \times 8$  shape layer. By filling the image with shapes in the codebook, it is divided into several shape regions. Each shape is marked with a different color, as shown on the right.

After the image is divided into shapes and their corresponding locations, it is necessary to convert them into binary data. We use natural binary coding for the first location and Golomb coding (Section 3.2.5) for the rest locations. Therefore, the binary data for each location are  $\{(x_i, y_i)_{bit} : 000000, 1010, 11111011, 1001, 11010, 1000, 1001 | m = 4\}$ . In training, we assign a one-to-one corresponding codeword to each shape in the codebook. There are seven shapes in Figure 8, two of which are the same. We need to find the codeword corresponding to each shape from the codebook, which is the binary data representation. For example, the result of combining each codeword and location may be  $\{((x_i, y_i)_{bit}, C_i) : (000000, 000), (1010, 001), (11111011, 010), (1001, 011), (11010, 100), (1000, 100), (1001, 101)\}$ . The second item in each bracket represents the codeword corresponding to the shape. Connecting all the small binary fragments is the binary data of the shape layer. On the other

hand, for clearly showing the compression ratio improvement, we use a complete image encoding example. Figure 9 illustrates its compression ratios with soft compression, PNG, JPEG2000 and JPEG-LS. The compression ratios are 2.53, 1.85, 2.41 and 2.45, respectively. This example indicates the improvement in the compression ratio of soft compression.



**Figure 9.** An example of image encoding with soft compression, PNG, JPEG2000 and JPEG-LS. It is a dermoscopic image from medical datasets. The compression ratios are 2.53, 1.85, 2.41 and 2.45, respectively.

### 3.3. Multi-Component Image

Considering a multi-component image, the soft compression algorithm for a gray image can be used for each component. In this case, the soft compression algorithm for a multi-component image is equivalent to the combination of several gray images. In addition, the compressed data are also a combination of several components.

## 4. Experimental Results and Theoretical Analysis

In this section, we reveal the experimental results and theoretical analysis of the soft compression algorithm for a binary image, gray image and multi-component image.

**Definition 5.** Suppose that  $b$  and  $b'$  represent the required number of bits to store the same image with natural binary code and other coding methods, respectively. The compression ratio  $R$  is defined as

$$R = \frac{b}{b'} \quad (51)$$

and  $R_{avg}$  is defined as the average compression ratio of a class of images.

The average compression ratio  $R_{avg}$  reflects the performance of different encoding methods. We adopt it as an significant criterion to measure the image compression algorithm.

### 4.1. Binary Image

We tested the soft compression algorithm for a binary image [1] on the MNIST [72] dataset. As expected, it had excellent results. In this subsection, we analyze the experimental results theoretically. The MNIST dataset has 10 classes. Each category of images may have a different compressible indicator value. Although they are of different classes, the compressible indicator value is generally subject to the normal distribution.

Table 1 illustrates the experimental results of the MNIST dataset with the soft compression algorithm for a binary image (each class uses the same codebook).  $CIV$  and  $R_{avg}$  are strongly related, and their Pearson correlation coefficient is 0.977. The larger the average compressible indicator value, the greater the average compression ratio. The results are consistent with the theoretical analysis in Theorem 2. It enlightens us to the fact that soft compression is suitable for compressing images with a large compressible indicator value. Although the compression ratio is not only determined by this factor, the compressible indicator value is the key element affecting the compression performance.

**Table 1.** Average compressible indicator value and compression ratio of MNIST dataset with soft compression algorithm for binary image (each class uses the same codebook).

Class	0	1	2	3	4	5	6	7	8	9
CIV	3.87	5.14	4.09	4.17	4.42	4.30	4.17	4.51	4.06	4.37
Compression ratio	<b>2.84</b>	<b>6.02</b>	<b>3.17</b>	<b>3.20</b>	<b>3.77</b>	<b>3.40</b>	<b>3.20</b>	<b>4.05</b>	<b>2.81</b>	<b>3.52</b>

#### 4.2. Gray Image and Multi-Component Image

In this subsection, we obtain the experimental results on different datasets with the soft compression algorithm. The compression ratio is one of the most important criteria to evaluate the performance of the encoding algorithm. Table 2 illustrates the average compression ratio of each class in Fashion-MNIST [73] with different methods. We emphasize the difference in percentage to soft compression for each other method in green if soft compression outperforms the other method. Our algorithm outperforms the widely-used PNG on all classes, which is a 24% improvement, on average. It also outperforms JPEG2000 and JPEG-LS, with 48% and 6.8% improvements, respectively.

Moreover, Table 3 shows the experimental results on datasets with larger images. The DRIVE [74] dataset is obtained from a diabetic retinopathy screening program to study skin lesions. PH2 [75] is a dermoscopic image database. PNG, JPEG2000 and JPEG-LS are the most popular methods in lossless image compression. L3C [46] is a novel learned lossless image compression system based on deep neural networks. We compare soft compression with these algorithms. In terms of the average compression ratio, soft compression outperforms any other method. On the other hand, soft compression and other methods have their own advantages for the maximum and minimum. The results of Tables 2 and 3 indicate that the soft compression algorithm is better than some known classical methods, PNG, JPEG2000, JPEG-LS and L3C, in terms of the compression ratio. For the same kind of image, it is better to choose soft compression to encode.

Table 4 illustrates the average compression ratio of the soft compression algorithm for a gray image on Fashion-MNIST. It is the result of cross validation. The first column denotes codebooks, which are generated by the training set of each class separately. The first row represents each category of the testing set. The value on  $(i, j)$  denotes the average compression ratio  $R_{avg}$  of the  $j$ -th class of the testing set by using the codebook generated by the  $i$ -th class of the training set. For example, the value on (Trouser, T-shirt) is the average compression ratio of T-shirt's testing set, whose codebook is trained on Trouser's training set. It is observed that the values on the diagonal are higher than those of the same column, which suggests that soft compression is related to the matching degree between the dataset and the codebook. The higher the matching degree, the higher the compression ratio.

**Table 2.** Average compression ratio of Fashion-MNIST with different image compression methods (images are gray and all methods are lossless compression).

Class	Method			
	Soft Compression	PNG	JPEG2000	JPEG-LS
T-Shirt	<b>1.53</b>	1.23 +24%	1.06 +44%	1.47 +4.1%
Trouser	<b>2.30</b>	1.50 +53%	1.32 +74%	2.13 +8.0%
Pullover	<b>1.48</b>	1.12 +32%	1.02 +45%	1.36 +8.8%
Dress	<b>1.85</b>	1.41 +31%	1.20 +54%	1.79 +3.4%
Coat	<b>1.45</b>	1.14 +27%	1.03 +41%	1.36 +6.7%
Sandals	<b>1.95</b>	1.82 +7.1%	1.33 +47%	1.82 +7.1%
Shirt	<b>1.42</b>	1.14 +25%	1.03 +38%	1.34 +6.0%
Sneaker	<b>2.07</b>	1.88 +10%	1.39 +49%	1.89 +9.5%
Bag	<b>1.50</b>	1.32 +14%	1.07 +40%	1.42 +5.6%
Ankle boots	<b>1.66</b>	1.46 +14%	1.14 +46%	1.52 +9.2%



**Table 3.** The compression results of DRIVE and PH2 datasets with different image compression methods (all methods are lossless compression). DRIVE is divided into training set and testing set according to the original division method. PH2 was not divided before, so the training set and testing set are divided in a 5:3 manner. L3C selects the best performing model among the provided models.

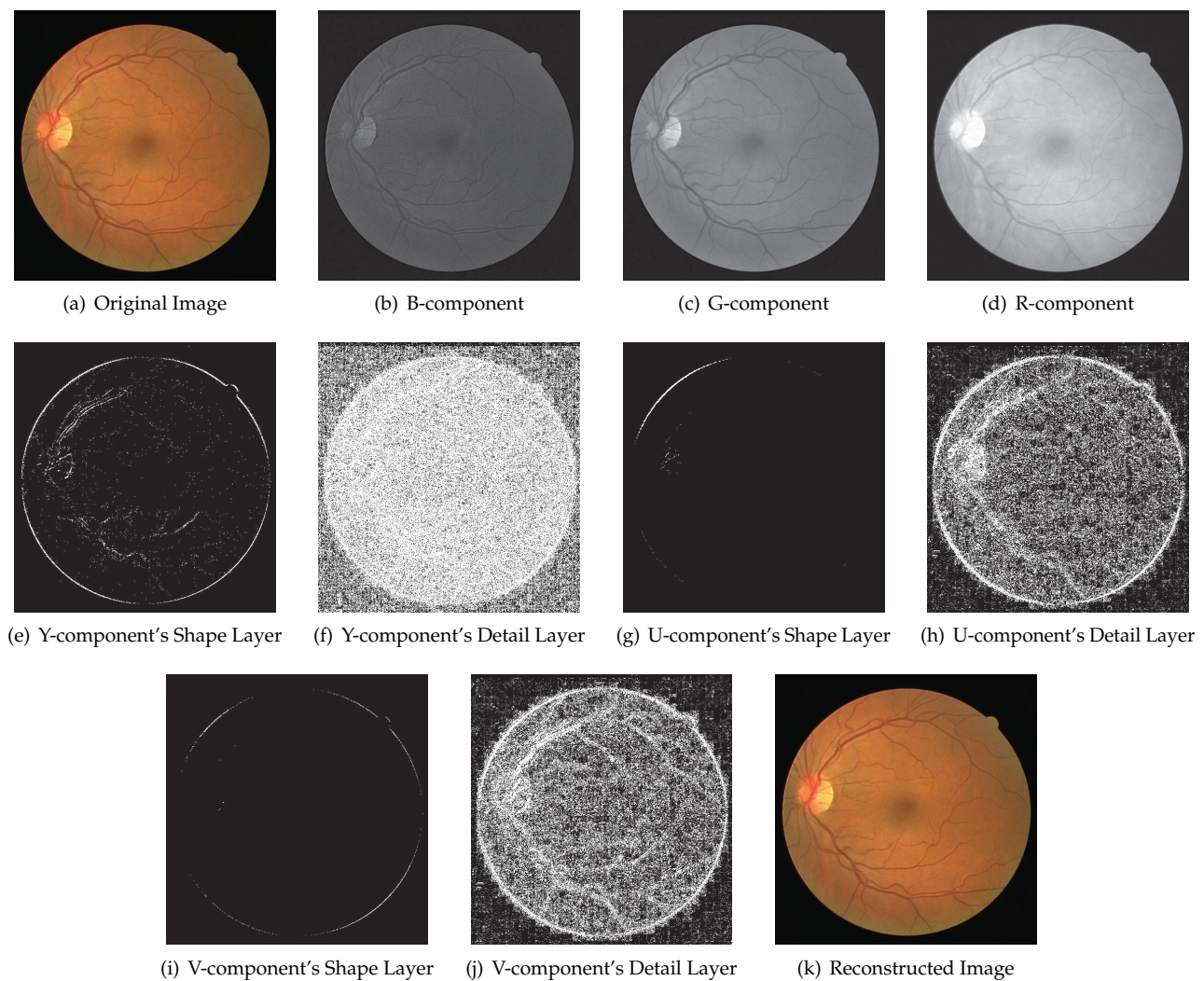
Dataset	Statistic	Method				
		Soft Compression	PNG	JPEG2000	JPEG-LS	L3C
DRIVE [74] 565 × 584 px	Mean	<b>3.201</b>	2.434 +32%	2.972 +7.7%	3.064 +4.5%	2.989 +7.1%
	Minimum	<b>2.893</b>	2.331 +24%	2.790 +3.7%	2.731 +5.9%	2.841 +1.8%
	Maximum	<b>4.171</b>	2.760 +51%	3.671 +14%	3.941 +5.8%	3.604 +16%
	Variance	0.0657	0.0072	0.0333	0.0632	0.0287
PH2 [75] 767 × 576 px	Mean	<b>2.570</b>	1.727 +49%	2.450 +4.9%	2.488 +3.3%	2.300 +12%
	Minimum	1.686	1.501	<b>1.812</b>	1.737	1.790
	Maximum	<b>3.388</b>	2.021 +68%	2.975 +14%	3.045 +11%	2.920 +16%
	Variance	0.1538	0.0108	0.0749	0.0835	0.1047

**Table 4.** Average compression ratio of Fashion-MNIST dataset with soft compression algorithm for gray image (each class has its own codebook).

Class	T-Shirt	Trouser	Pullover	Dress	Coat	Sandals	Shirt	Sneaker	Bag	Ankle Boots
T-shirt	1.55	2.19	1.50	1.83	1.48	1.90	1.44	2.00	1.51	1.65
Trouser	1.48	2.35	1.43	1.82	1.41	1.91	1.38	2.03	1.46	1.61
Pullover	1.55	2.20	1.50	1.82	1.48	1.88	1.44	1.99	1.51	1.65
Dress	1.54	2.32	1.48	1.87	1.46	1.96	1.43	2.08	1.51	1.66
Coat	1.54	2.20	1.50	1.83	1.48	1.88	1.44	2.00	1.51	1.65
Sandals	1.53	2.27	1.47	1.85	1.45	2.01	1.42	2.11	1.51	1.68
Shirt	1.55	2.20	1.50	1.83	1.48	1.89	1.44	2.00	1.51	1.65
Sneaker	1.52	2.27	1.46	1.84	1.45	1.99	1.41	2.11	1.51	1.67
Bag	1.55	2.25	1.49	1.84	1.47	1.94	1.44	2.06	1.53	1.67
Ankle boots	1.54	2.24	1.49	1.84	1.47	1.94	1.43	2.06	1.51	1.67

In practice, it is better to adopt a corresponding codebook for the specific category of images. However, imperfect matching may reduce the compression ratio, but will not cause any loss of information. In fact, codebooks of soft compression are complete. That is to say that for any codebook and any picture, lossless compression can always be achieved. The difference lies in diverse compression ratios. From Table 4, we can draw a conclusion that the compression ratio is both related to images and codebooks. This corresponds to the compressible indicator value and the similarity between images and codebooks. The larger the compressible indicator value, the higher the similarity between images and codebooks, and the higher the compression ratio.

For a multi-component image, its process is the combination of several gray images. Figure 10 illustrates an example with the soft compression algorithm for a multi-component image. Subfigure (a) is a multi-component original image, whose components are B, G and R, respectively, as shown in (b), (c) and (d). After dividing each component into the shape layer and detail layer, respectively (binarization was made for a clearer appearance), one can obtain subfigures (e) to (j). Furthermore, the compressed file can be generated by adopting the above-mentioned encoding method for the shape layer and detail layer. The reconstructed subfigure (k) can be obtained through decoding, which is the same as the original subfigure (a).



**Figure 10.** An example of DRIVE dataset with soft compression algorithm for multi-component image. (a) The original RGB image. (b) B-component image obtained by separation. (c) G-component image obtained by separation. (d) R-component image obtained by separation. (e) The shape layer (binarization) of Y-component. (f) The detail layer (binarization) of Y-component. (g) The shape layer (binarization) of U-component. (h) The detail layer (binarization) of U-component. (i) The shape layer (binarization) of V-component. (j) The detail layer (binarization) of V-component. (k) The reconstructed image.

#### 4.3. Implementation Details

The architecture of the soft compression algorithm is shown in Figure 2. Detailed implementation can be found in the code. The algorithm is implemented with Python on a single Intel i7-9700K CPU. The batch size is 10 for Fashion-MNIST and 1 for the other two datasets. In addition, the shape degree is set to 0.1 for Fashion-MNIST and 0.5 for the other three datasets.

The encoding and decoding complexity are both related to the image size and the number of shapes in the codebook, i.e.,  $\mathcal{O}(MN|S|)$ . Moreover, the average encoding and decoding times of an image are shown in Table 5.

**Table 5.** Average encoding and decoding times of images with soft compression algorithm.

	Fashion-MNIST [73] 28 × 28 px	DRIVE [74] 565 × 584 px	PH2 [75] 767 × 576 px
Encoding	$5.7 \times 10^{-2}$ s	7.79 s	24.98 s
Decoding	$4.1 \times 10^{-3}$ s	5.31 s	6.80 s

## 5. Conclusions

In this paper, we investigated how to apply soft compression to encode images in the lossless mode. It uses shapes to encode an image, which aims to eliminate coding redundancy and spatial redundancy at the same time. Due to the adaptability and completeness of codebooks with soft compression, it can always achieve lossless compression for any image and any codebook.

In theory, we also proposed a new concept, the compressible indicator function with regard to images, and theoretically illustrated the working principle. The compressible indicator function points out the suitable scenarios of soft compression. In addition, it also gives a threshold about the required number of bits to represent a location with soft compression.

Moreover, we designed soft compression algorithms for a binary image, gray image and multi-component image. These algorithms were tested on the datasets. Experimental results indicated that soft compression has significant effects on lossless image compression, which outperform classical systems PNG and JPEG2000, especially for images which have a large compressible indicator value.

This paper focuses on lossless compression. However, soft compression can also be combined with other transformation methods, such as wavelet transform. Lossy compression can be realized by using soft compression for the coefficients of transform domain. Soft compression can also be combined with channel coding to enhance the effect of joint source-channel coding.

It is expected that this work may have excellent applications when errors cannot be tolerated or where there is critical social or scientific value, such as CT image processing for the diagnosis and treatment of medical image files, digital libraries and so on.

**Author Contributions:** Conceptualization, P.F. and G.X.; methodology, P.F. and G.X.; software, G.X.; formal analysis, P.F. and G.X.; writing—original draft preparation, G.X.; writing—review and editing, P.F. and G.X.; visualization, P.F. and G.X.; supervision, P.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Beijing Natural Science Foundation No. 4502030.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The code of soft compression for gray image is available: <https://github.com/ten22one/Soft-compression-algorithm-for-gray-image> (accessed on 9 December 2021).

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

## References

1. Xin, G.; Li, Z.; Zhu, Z.; Wan, S.; Fan, P.; Letaief, K.B. Soft Compression: An Approach to Shape Coding for Images. *IEEE Commun. Lett.* **2020**, *25*, 798–801. [[CrossRef](#)]
2. Gonzalez, R.C.; Woods, R.E.; Masters, B.R. *Digital Image Processing*, 3rd ed.; Pearson Prentice Hall: Hoboken, NJ, USA, 2008; pp. 527–553.
3. Huffman, D.A. A method for the construction of minimum-redundancy codes. *Proc. IRE* **1952**, *40*, 1098–1101. [[CrossRef](#)]
4. Rissanen, J.; Langdon, G.G. Arithmetic coding. *IBM J. Res. Dev.* **1979**, *23*, 149–162. [[CrossRef](#)]
5. Golomb, S. Run-length encodings (corresp.). *IEEE Trans. Inf. Theory* **1966**, *12*, 399–401. [[CrossRef](#)]
6. Meyr, H.; Rosdolsky, H.; Huang, T. Optimum run length codes. *IEEE Trans. Commun.* **1974**, *22*, 826–835. [[CrossRef](#)]
7. Welch, T.A. A technique for high-performance data compression. *Computer* **1984**, *17*, pp. 8–19. [[CrossRef](#)]
8. Starosolski, R. Hybrid adaptive lossless image compression based on discrete wavelet transform. *Entropy* **2020**, *22*, 751. [[CrossRef](#)]
9. Cao, S.; Wu, C.Y.; Krähenbühl, P. Lossless image compression through super-resolution. *arXiv* **2020**, arXiv:2004.02872.
10. Consultative Committee for Space Data Systems. Available online: <https://www.ccsds.org/> (accessed on 9 December 2021).
11. Blanes, I.; Magli, E.; Serra-Sagrasta, J. A tutorial on image compression for optical space imaging systems. *IEEE Geosci. Remote Sens. Mag.* **2014**, *2*, 8–26. [[CrossRef](#)]
12. Augé, E.; Sánchez, J.E.; Kiely, A.B.; Blanes, I.; Serra-Sagrasta, J. Performance impact of parameter tuning on the CCSDS-123 lossless multi-and hyperspectral image compression standard. *J. Appl. Remote Sens.* **2013**, *7*, 074594. [[CrossRef](#)]

13. Sezer, O.G.; Harmanci, O.; Guleryuz, O.G. Sparse orthonormal transforms for image compression. In Proceedings of the 2008 15th IEEE International Conference on Image Processing, San Diego, CA, USA, 12–15 October 2008; pp. 149–152.
14. Zhou, Y.; Wang, C.; Zhou, X. DCT-based color image compression algorithm using an efficient lossless encoder. In Proceedings of the 2018 14th IEEE International Conference on Signal Processing (ICSP), Beijing, China, 2–16 August 2018; pp. 450–454.
15. Ma, H.; Liu, D.; Yan, N.; Li, H.; Wu, F. End-to-End Optimized Versatile Image Compression With Wavelet-Like Transform. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [[CrossRef](#)]
16. Antonini, M.; Barlaud, M.; Mathieu, P.; Daubechies, I. Image coding using wavelet transform. *IEEE Trans. Image Process.* **1992**, *1*, 205–220. [[CrossRef](#)]
17. Remya, S. Wavelet Based Compression Techniques: A Survey. In Proceedings of the International Conference on Advances in Communication, Network, and Computing, Chennai, India, 24–25 February 2012; pp. 394–397.
18. Starosolski, R. Employing New Hybrid Adaptive Wavelet-Based Transform and Histogram Packing to Improve JP3D Compression of Volumetric Medical Images. *Entropy* **2020**, *22*, 1385. [[CrossRef](#)] [[PubMed](#)]
19. Kumar, R.N.; Jagadale, B.; Bhat, J. A lossless image compression algorithm using wavelets and fractional Fourier transform. *SN Appl. Sci.* **2019**, *1*, 1–8.
20. Lin, J. Reversible Integer-to-Integer Wavelet Filter Design for Lossless Image Compression. *IEEE Access* **2020**, *8*, 89117–89129. [[CrossRef](#)]
21. Kabir, M.; Mondal, M. Edge-based and prediction-based transformations for lossless image compression. *J. Imaging* **2018**, *4*, 64. [[CrossRef](#)]
22. Punnappurath, A.; Brown, M.S. Learning raw image reconstruction-aware deep image compressors. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *42*, 1013–1019. [[CrossRef](#)] [[PubMed](#)]
23. Yang, Y.; Sun, J.; Li, H.; Xu, Z. ADMM-CSNet: A deep learning approach for image compressive sensing. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *42*, 521–538. [[CrossRef](#)]
24. Zhang, X.; Wu, X. Attention-guided Image Compression by Deep Reconstruction of Compressive Sensed Saliency Skeleton. *arXiv* **2021**, arXiv:2103.15368.
25. Ballé, J.; Laparra, V.; Simoncelli, E.P. End-to-end optimized image compression. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.
26. Theis, L.; Shi, W.; Cunningham, A.; Huszár, F. Lossy image compression with compressive autoencoders. *arXiv* **2017**, arXiv:1703.00395.
27. Baig, M.H.; Koltun, V.; Torresani, L. Learning to inpaint for image compression. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1246–1255.
28. Rippel, O.; Bourdev, L. Real-time adaptive image compression. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2922–2930.
29. Ballé, J.; Minnen, D.; Singh, S.; Hwang, S.J.; Johnston, N. Variational image compression with a scale hyperprior. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
30. Toderici, G.; Vincent, D.; Johnston, N.; Hwang, S.J.; Minnen, D.; Shor, J.; Covell, M. Full Resolution Image Compression with Recurrent Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5435–5443.
31. Agustsson, E.; Tschannen, M.; Mentzer, F.; Timofte, R.; Gool, L.V. Generative adversarial networks for extreme learned image compression. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, South Korea, 27 October–2 November 2019; pp. 221–231.
32. Minnen, D.; Ballé, J.; Toderici, G. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. In Proceedings of the NeurIPS 2018, Montreal, QC, Canada, 3–8 December 2018.
33. Lee, J.; Cho, S.; Beack, S.K. Context-adaptive Entropy Model for End-to-end Optimized Image Compression. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
34. He, D.; Zheng, Y.; Sun, B.; Wang, Y.; Qin, H. Checkerboard Context Model for Efficient Learned Image Compression. *arXiv* **2021**, arXiv:2103.15306.
35. Gulrajani, I.; Kumar, K.; Ahmed, F.; Taiga, A.A.; Visin, F.; Vazquez, D.; Courville, A. Pixelvae: A latent variable model for natural images. *arXiv* **2016**, arXiv:1611.05013.
36. Chen, X.; Mishra, N.; Rohaninejad, M.; Abbeel, P. Pixelsnail: An improved autoregressive generative model. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 864–872.
37. Mentzer, F.; Gool, L.V.; Tschannen, M. Learning better lossless compression using lossy compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6638–6647.
38. Li, M.; Ma, K.; You, J.; Zhang, D.; Zuo, W. Efficient and Effective Context-Based Convolutional Entropy Modeling for Image Compression. *IEEE Trans. Image Process.* **2020**, *29*, 5900–5911. [[CrossRef](#)]
39. Van Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel recurrent neural networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1747–1756.
40. Oord, A.v.d.; Kalchbrenner, N.; Vinyals, O.; Espeholt, L.; Graves, A.; Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. *arXiv* **2016**, arXiv:1606.05328.

41. Salimans, T.; Karpathy, A.; Chen, X.; Kingma, D.P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv* **2017**, arXiv:1701.05517.
42. Kingma, F.; Abbeel, P.; Ho, J. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 3408–3417.
43. Townsend, J.; Bird, T.; Barber, D. Practical lossless compression with latent variables using bits back coding. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
44. Hoogeboom, E.; Peters, J.W.; Berg, R.V.d.; Welling, M. Integer discrete flows and lossless compression. *arXiv* **2019**, arXiv:1905.07376.
45. Ho, J.; Lohn, E.; Abbeel, P. Compression with flows via local bits-back coding. *arXiv* **2019**, arXiv:1905.08500.
46. Mentzer, F.; Agustsson, E.; Tschannen, M.; Timofte, R.; Gool, L.V. Practical full resolution learned lossless image compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 10629–10638.
47. Zhu, C.; Zhang, H.; Tang, Y. Lossless Image Compression Algorithm Based on Long Short-term Memory Neural Network. In Proceedings of the 2020 5th International Conference on Computational Intelligence and Applications (ICCIA), Beijing, China, 19–21 June 2020; pp. 82–88.
48. Boutell, T.; Lane, T. PNG (Portable Network Graphics) Specification Version 1.0. Available online: <https://www.hjp.at/doc/rfc/rfc2083.html> (accessed on 9 December 2021).
49. Dufaux, F.; Sullivan, G.J.; Ebrahimi, T. The JPEG XR image coding standard [Standards in a Nutshell]. *IEEE Signal Process Mag.* **2009**, *26*, 195–204. [[CrossRef](#)]
50. Weinberger, M.J.; Seroussi, G.; Sapiro, G. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Trans. Image Process.* **2000**, *9*, 1309–1324. [[CrossRef](#)] [[PubMed](#)]
51. Si, Z.; Shen, K. Research on the WebP image format. In *Advanced Graphic Communications, Packaging Technology and Materials*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 271–277.
52. Wallace, G.K. The JPEG still picture compression standard. *IEEE Trans. Consum. Electron.* **1992**, *38*, 18–34. [[CrossRef](#)]
53. Skodras, A.; Christopoulos, C.; Ebrahimi, T. The jpeg 2000 still image compression standard. *IEEE Signal Process Mag.* **2001**, *18*, 36–58. [[CrossRef](#)]
54. Ahmed, N.; Natarajan, T.; Rao, K.R. Discrete cosine transform. *IEEE Trans. Comput.* **1974**, *100*, 90–93. [[CrossRef](#)]
55. Mallat, S.G. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **1989**, *11*, 674–693. [[CrossRef](#)]
56. Sneyers, J.; Wuille, P. FLIF: Free lossless image format based on MANIAC compression. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 66–70.
57. Ascher, R.N.; Nagy, G. A means for achieving a high degree of compaction on scan-digitized printed text. *IEEE Trans. Comput.* **1974**, *100*, 1174–1179. [[CrossRef](#)]
58. Shen, Z.; Frater, M.; Arnold, J. Optimal Pruning Quad-Tree Block-Based Binary Shape Coding. In Proceedings of the 2007 IEEE International Conference on Image Processing, San Antonio, TX, USA, 16 September–19 October 2007; pp. 437–440.
59. Shu, Z.; Liu, G.; Xie, Z.; Ren, Z. Shape adaptive texture coding based on wavelet-based contourlet transform. In Proceedings of the 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Beijing, China, 13–15 October 2018; pp. 1–5.
60. Jacquin, A.E. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. Image Process.* **1992**, *1*, 18–30. [[CrossRef](#)] [[PubMed](#)]
61. Weinzaepfel, P.; Jégou, H.; Pérez, P. Reconstructing an image from its local descriptors. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 337–344.
62. Yue, H.; Sun, X.; Wu, F.; Yang, J. SIFT-based image compression. In Proceedings of the 2012 IEEE International Conference on Multimedia and Expo, Melbourne, Australia, 9–13 July 2012; pp. 473–478.
63. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **2004**, *60*, 91–110. [[CrossRef](#)]
64. Yue, H.; Sun, X.; Yang, J.; Wu, F. Cloud-Based Image Coding for Mobile Devices—Toward Thousands to One Compression. *IEEE Trans. Multimed.* **2013**, *15*, 845–857. [[CrossRef](#)]
65. Bégaint, J.; Thoreau, D.; Guillotel, P.; Guillemot, C. Region-Based Prediction for Image Compression in the Cloud. *IEEE Trans. Image Process.* **2018**, *27*, 1835–1846. [[CrossRef](#)]
66. Liu, X.; Cheung, G.; Lin, C.; Zhao, D.; Gao, W. Prior-Based Quantization Bin Matching for Cloud Storage of JPEG Images. *IEEE Trans. Image Process.* **2018**, *27*, 3222–3235. [[CrossRef](#)]
67. Zhang, X.; Lin, W.; Zhang, Y.; Wang, S.; Ma, S.; Duan, L.; Gao, W. Rate-Distortion Optimized Sparse Coding With Ordered Dictionary for Image Set Compression. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 3387–3397. [[CrossRef](#)]
68. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [[CrossRef](#)]
69. Cover, T.M. *Elements of Information Theory*; John Wiley & Sons: Hoboken, NJ, USA, 1999.
70. Wu, X.; Memon, N. Context-based, adaptive, lossless image coding. *IEEE Trans. Commun.* **1997**, *45*, 437–444. [[CrossRef](#)]
71. Schwartz, J.W.; Barker, R.C. Bit-Plane Encoding: A Technique for Source Encoding. *IEEE Trans. Aerosp. Electron. Syst.* **1966**, *AES-2*, 385–392. [[CrossRef](#)]

- 
72. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
  73. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
  74. DRIVE: Digital Retinal Images for Vessel Extraction. Available online: <https://drive.grand-challenge.org/> (accessed on 9 December 2021).
  75. Mendonça, T.; Ferreira, P.M.; Marques, J.S.; Marcal, A.R.; Rozeira, J. PH 2-A dermoscopic image database for research and benchmarking. In Proceedings of the 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Osaka, Japan, 3–7 July 2013; pp. 5437–5440.