IMAGE COMPRESSION FOR PRECISION LIVESTOCK FARMING

Jairo Miguel Marulanda Giraldo
Universidad Eafit Country
jmmarulang@eafit.edu.co

Sebastian Herrera Beleño
Universidad Eafit
Colombia
sherrerab1@eafit.edu.co

Simón Marín
Universidad Eafit
Colombia
smaring1@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

ABSTRACT

The desire for quantitative, qualitative, individual tracing of livestock in the agricultural sector has created the need for more efficient data analysis, including image processing. Of this development depends the formation of a truly efficient and useful technical farming sector.

Image compression can be applied to many problems, methods have also been investigated for more classical interpolation problems such as zooming into an image by increasing its resolution.

Which is the algorithm you proposed? What results did you achieve? What are the conclusions of this work? The abstract should have **at most 200 words**. (*In this semester, you should summarize here execution times, memory consumption, compression ratio and accurracy*).

Keywords

Compression algorithms, machine learning, deep learning, precision livestock farming, animal health.

1. INTRODUCTION

Efficient algorithms for image compression always be a problem planted in all solutions that involve analysis, implementation, or image manipulation.

As an example, when a service stores a lot of images, theses need to be compressed when they are stored in servers this aiming to reduce the size of the images and the space needed to save them, reducing cost of infrastructure.

An efficient algorithm of image compression in the context of precision livestock farming allows to reduce costs and the accessibility to farms for implements software that help them to increase the lifetime of cattle.

1.1. Problem

The problem we confront is the need for efficient image compression algorithms, developed for the growing Precision Livestock farming sector.

With growing population, and hence food demands, the need for a more efficient agricultural industry is rapidly intensifying. For with higher demand comes less time available for production.

Additionally, in Colombia, there are 14 different unofficially controlled diseases which highly decrease productivity, animal reproductivity and the stability of the industry. Including, for example, Tuberculosis, brucellosis, bovine rabies and foot and mouth disease. [8]

Efficient image processing, focused on livestock health, could have a radical impact in overcoming these challenges.

1.2 Solution

In this work, we used a convolutional neural network to classify animal health, in cattle, in the context of precision livestock farming (PLF). A common problem in PLF is that networking infrastructure is very limited, thus data compression is required.

We wanted to explore different solutions for this problem, and Annalise conveniences for each one. With Nearest Neighbors we can appreciate a big loss of information and in some cases its impossible recognize the picture, special in small dimensions. For these reasons we decided use linear interpolation for the compression of data, even if the disk size of the image is not reduced as in Nearest Neighbors. We want to maintain the effectiveness of the model and not generate a counterproductive effect.

1.3 Article structure

In what follows, in Section 2, we present related work to the problem. Later, in Section 3, we present the data sets and methods used in this research. In Section 4, we present the algorithm design. After, in Section 5, we present the results. Finally, in Section 6, we discuss the results and we propose some future work directions.

2. RELATED WORK

In what follows, we explain four related works on the domain of animal-health classification and image compression in the context of PLF.

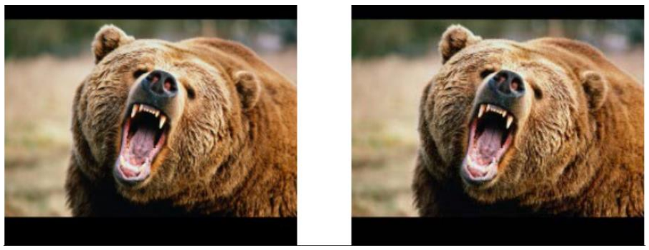
2.1 High-Performance System for Secure Image Communication in the Internet of Things

They create the BPG format which has many advantages over JPEG. It achieves a higher compression ratio with smaller size than JPEG for similar quality. The algorithm can be divided into 4 functions: BPG encoder, BPG decoder, JavaScript decoder and BPG decoding. The first takes the image in JPEG or PNG as input, performs BPG compression and returns the image in BPG format. The need for an algorithm supported by most browsers is a problem that BPG solves. [13]

The BPG encoder is based on HEVG encoding which is considered the greatest advance in compression techniques. HEVC offers great efficiency due to the smart approach used to reduce the area in pixels that is being encoded.

Some of the metrics used for this compression algorithm are:

- RMSE: The average difference of squares, pixel by pixel
- PNSR: Light component
- SSIM: Correlated with human perception: Brightness, contrast and structure.
- MSSIM: Difference and Cross Correlation
- VIF: Mutual information



[13]

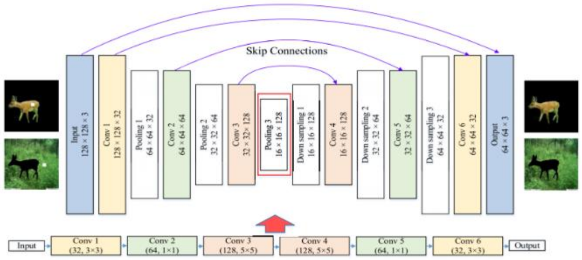
Test Image	Compression	Size (KB)	RMSE	SSIM	VIF	MSSIM	PSNR
Bear Image	JPEG (input image)	19.4	0.015	0.960	5.097	8.691	84.0
	BPG image	15.8	0.012	0.977	5.201	9.008	84.8

2.2 High-Efficiency Progressive Transmission and Automatic Recognition of Wildlife Monitoring Images with WISNs

Wireless Image Sensor Networks (WISN) are widely applied in wildlife monitoring, as they perform better in real-time remote monitoring. Traditional WISNs suffer from the limitations of low throughput, power consumption restrictions, and narrow transmission bandwidth. [14]

A convolutional codec was used to prioritize the transmission of the important region and ensure that staff can receive the region of greatest interest for the first time. [14]

They selected 1,000 images as a training set, 100 images as a validation set, and 200 images as a test set. All images are 128 × 128 pixels in size. The following figure uses the ADAM optimizer. [14]



For loss of training, they use the root mean square error (MSE) between the restored images as follows, which indicates the degree of inconsistency of the estimated value. Compared to the cross-entropy function, this loss function is related to the correct and incorrect prediction result to make the incorrect result become an average value.

2.3 Estimation of Primary Quantization Steps in Double-Compressed JPEG Images Using a Statistical Model of Discrete Cosine Transform

Double compression of images occurs when one compresses twice, possibly with different quality factors, a digital image.

Estimation of the first compression parameter of such a double compression is of a crucial interest for image forensics since it may help revealing, for instance, the software or the source camera. This paper proposes an accurate method for estimating the primary quantization steps in double-compressed JPEG images. [15]

In contrast with prior statistical model-based methods, the paper proposes to exploit the state-of-the-art statistical model of once-quantized DCT coefficients. This model can accurately capture statistics of DCT coefficients, which leads to improve considerably the estimation accuracy. [15]

The article proposes a method for the estimation of quantization steps (of pixel rounding) using a model of secondary DCT coefficients.

3. MATERIALS AND METHODS

In this section, we explain how the data was collected and processed and, after, different image-compression algorithm alternatives to solve improve animal-health classification.

3.1 Data Collection and Processing

We collected data from Google Images and Bing Images divided into two groups: healthy cattle and sick cattle. For healthy cattle, the search string was “cow”. For sick cattle, the search string was “cow + sick”.

In the next step, both groups of images were transformed into grayscale using Python OpenCV and they were transformed into Comma Separated Values (CSV) files. It was found out that the datasets were balanced.

The dataset was divided into 70% for training and 30% for testing. Datasets are available at https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets.

Finally, using the training data set, we trained a convolutional neural network for binary image-classification using Google Teachable Machine available at https://teachablemachine.withgoogle.com/train/image.

3.2 Lossy Image-compression alternatives

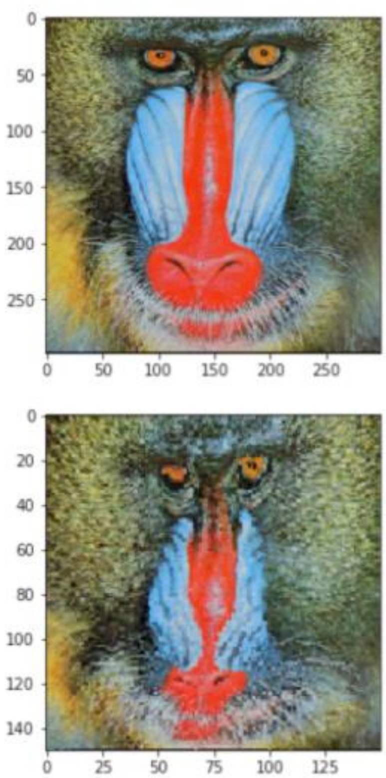
In what follows, we present different algorithms used to compress images. *(In this semester, examples of such* *algorithms are Seam carving, image scaling, discrete cosine transform, wavelet compression and fractal compression).*

3.2.1 Seam Carving

This technique was invented on the 2007 paper by Shai Avidan and Ariel Shamir "Seam Carving for Content-Aware Image Resizing".

A seam is defined as an optimal 8-connected path of pixels, that is, that are connected to each other sharing an edge or a corner. Seams range from either the top of an image to its bottom or from side to side, contain one pixel per row or column (depending on their orientation) and their path is defined by an energy function. The output of this functions can be determined by several mathematical functions They are, however, created with the underlying idea that, rapid change on pixel intensity indicates a more relevant part of the image, and hence will have higher energy. [6] [10]

The algorithm then removes one seam at a time, starting with the ones with the lowest energy value, until the desire compression is achieved. [6] [10] Additionally, Seam Carving can be used to resize an image to any size, not just compression. [6] [10]

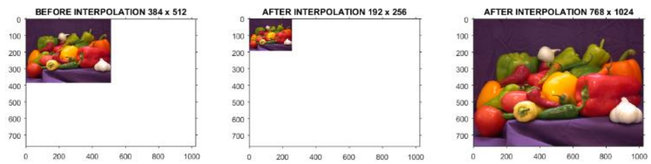


[6]

3.2.2 Nearest-neighbor interpolation

This algorithm is one of the simplest ways of resizing an image. This method, when resizing, simply selects a neighboring pixel´s and assumes is intensity value. [1] [2] [3]

It is commonly used in real-time 3d rendering, as in the zoom tool in Photoshop. [1]



[3]

3.2.3 Discrete Cosine Transform (DCT)

This algorithm takes the spatial domain of an image, as in it takes the intensity values of pixel clusters in the picture, to transform it into a frequency domain. This final domain is determined by expressing the image as a two-dimensional matrix and then applying the DCT formula (shown
below). [10]

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p=0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q=0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

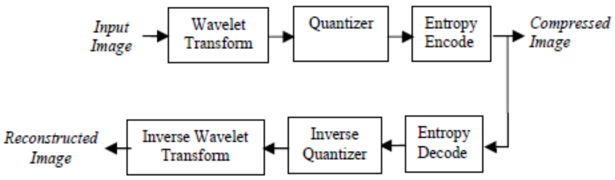
This formula essentially searches for a superposition of cosines whose outputs better represent the original pixel clusters. Then quantization is applied, that is discrete values are selected, and redundant data is removed. [10]

To get the original image back, the inverse formula must be applied. Because of the quantization process, the quality of reconstructed image and compression ratio is improved. [10]

3.2.4 Discrete Wavelet Transform (DWT)

Wavelets are signals which are local in time and scale, with generally an irregular shape, who can separate the fine details in a larger signal, like the ones given by an image color intensity. They can be used to decompose them into components. [10]

DWT uses wavelets to decompose images into four sub-images, denominated LL, LH, HL and HH. Each of these sub-images extracts different features of the original. Of this, the most relevant is LL, which can take care of fine detail and so is used for further decomposition. Then quantization is applied to get wavelet coefficients and the entropy coding is applied on wavelet coefficients to get the compressed image data. [10] [12]

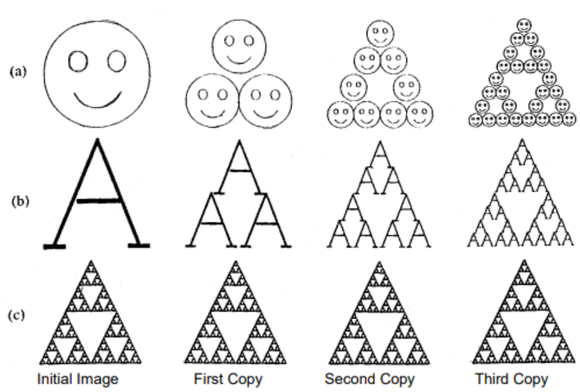


[12]

3.2.4 Fractal Encoding

Fractal identifies redundant data in an image and removes it to be compressed. [10]

It functions by dividing the image into non-overlapping blocks and then, by mathematical methods comparing them between each other. If the blocks do not surpass an error tolerance of similarity, the information is recorded. Otherwise, the blocks are subdivided and the whole process repeated. [10]



3.3 Lossless Image-compression alternatives

In what follows, we present different algorithms used to compress images. *(In this semester, examples of such algorithms are Borrows & Wheeler Transform, LZ77, LZ78, Huffman coding and LZS).*

3.3.1 Burrows-Wheeler Transform (BWT)

This algorithm was invented by Michael Burrows and David Wheeler in 1994. It is commonly used in data compression, as in bzip2. [9]

BWT begins by creating a table with all possible permutations of a given string. It then rearranges alphabetically and returns the last column of the table. This final column can be later be used as a base to reconstruct the whole string without data loss. [7]

<table>
<tr><th colspan="5">Transformation</th></tr>
<tr><th>Input</th><th>All Rotations</th><th>Sorting All Rows into Lex Order</th><th>Taking Last Column</th><th>Output Last Column</th></tr>
<tr><td>^BANANA|</td><td>^BANANA|
|^BANANA
A|^BANAN
NA|^BANA
ANA|^BAN
NANA|^BA
ANANA|^B
BANANA|^</td><td>ANANA|^B
ANA|^BAN
A|^BANAN
BANANA|^
NANA|^BA
NA|^BANA
^BANANA|
|^BANANA</td><td>ANANA|^B
ANA|^BAN
A|^BANAN
BANANA|^
NANA|^BA
NA|^BANA
^BANANA|
|^BANANA</td><td>BNN^AA|A</td></tr>
</table>

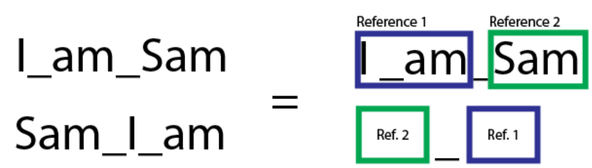
3.3.2 LZ77, LZ78 and LZSS

LZ77 and LZ78 are dictionary-based lossless schemes. These compressors replace sub-strings in a dataset with shorter codewords, stored in dictionaries. [11]

In LZ77 the codewords are composed in base of the most recently encoded data, also called search buffer. These code-words are pointers to the longest match in the buffer for a given sub-string. [11]

The LZ77 approach needs the patterns to occur close to each other to be efficient. The ZL78 does not have this need, for it uses not a buffer. Instead, the dictionary in this scheme is an indexed list of some previously encountered sub-strings. [11]

Finally, we have the LZSS scheme, which was derived from LZ77. This scheme has two main differences from its predecessor. First, unlike LZ77, LZSS does not permit the dictionary reference to be longer that the sub-string it is replacing. Second, it uses one-bit flags to indicate whether the next chunk of data is a literal (byte) or a reference to an offset/length pair. [5]



[11]

3.3.4 Huffman Encoding

A variable length coding technique. It identifies frequently occurring patterns in a data set and assigns them a code word. The more frequent the pattern, the shorter the word assigned. [11]

For this, the algorithm uses a binary tree, constructer from bottom up, where the leaf nodes represent the patterns and their corresponding probability, located in descending order. It then adds the lower probabilities and repeats this process until two or less probabilities are left. This generates the binary tree. Finally, the algorithm assigns the corresponding code words and saves the data. [11]

4. ALGORITHM DESIGN AND IMPLEMENTATION

In what follows, we explain the data structures and the algorithms used in this work. The implementations of the data structures and algorithms are available at Github[1].

4.1 Data Structures

We use a 2d Array to represent the image in their 2 components x and y. Each position of the array represents the intensity of the corresponding pixel. This facilitates the access to the pixels.

	0	1	2	3	4
0	245	245	245	245	245
1	230	220	240	200	192
2	220	214	219	234	193
3	212	200	209	194	197

Figure 1: 2D array with the 20 first pixels of an image.

4.2 Algorithms

In this work, we propose a compression algorithm which is a combination of a lossy image-compression algorithm and a lossless image-compression algorithm. We also explain how decompression for the proposed algorithm works.

We implemented linear interpolation with 2 variants: With scaling and without scaling of dimensions of the image. In some cases, the image is needlessly large, for these cases we need to use linear interpolation with scaling.

4.2.1 Linear interpolation with scaling

To calculate the value of each new pixel of the compressed image we take a block of 4 pixels from the original image and get its average value (intensity), the result will be saved in a new 2d array and all the four blocks receive this new average value.

$$p[y][x] = \frac{p[y][x] + p[y+1][x] + p[y][x+1] + p[y+1][x+1]}{4}$$

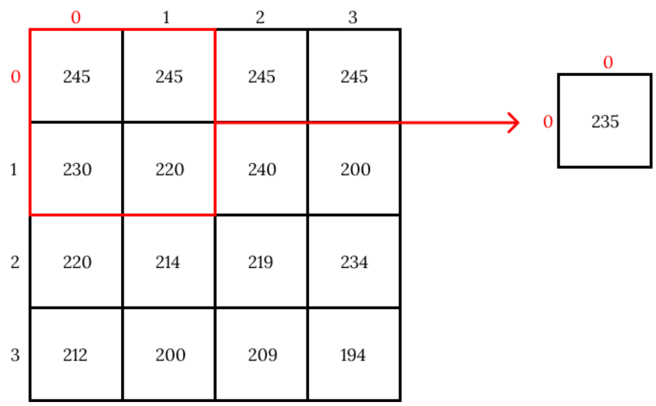


Figure 2: The formula used to calculate each new pixel value.

Figure 3: The graphic method of linear interpolation with scaling

4.2.2 Linear interpolation whiteout scaling

The method is the same, we take a block of 4 pixel from the original image and get the average. But unlike of the scaling method we save the result in the same position from the original image in the new compress image.

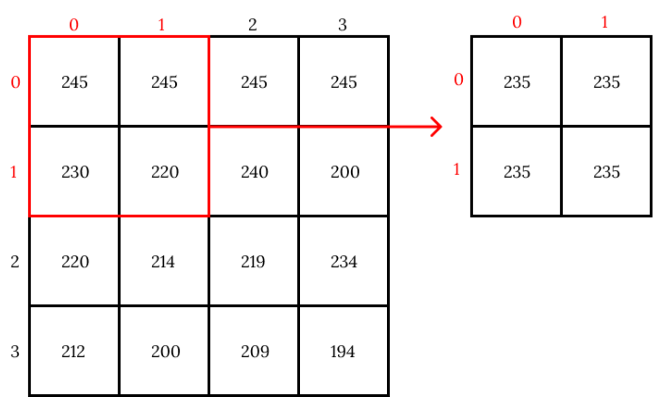


Figure 3: The graphic method of linear interpolation without scaling

4.3 Complexity analysis of the algorithms

[1]https://github.com/sebasbeleno/ST0245-001

Explain, in your own words, the analysis for the worst case using O notation. How did you calculate such complexities. Please explain briefly.

Algorithm	**Time Complexity**
Compression	$O(N^2 * M^2)$
Decompression	$O(N^3 * M * 2^N)$

Table 2: Time Complexity of the image-compression and image-decompression algorithms. *(Please explain what do N and M mean in this problem).*

Algorithm	**Memory Complexity**
Compression	$O(N * M * 2^N)$
Decompression	$O(2^{M*} 2^N)$

Table 3: Memory Complexity of the image-compression and image-decompression algorithms. *(Please explain what do N and M mean in this problem).*

4.4 Design criteria of the algorithm

Explain why the algorithm was designed that way. Use objective criteria. Objective criteria are based on efficiency, which is measured in terms of time and memory consumption. Examples of non-objective criteria are: "I was sick", "it was the first data structure that I found on the Internet", "I did it on the last day before deadline", etc. Remember: This is 40% of the project grading.

5. RESULTS

5.1 Model evaluation

In this section, we present some metrics to evaluate the model. Accuracy is the ratio of number of correct predictions to the total number of input samples. Precision. is the ratio of successful students identified correctly by the model to successful students identified by the model. Finally, Recall is the ratio of successful students identified correctly by the model to successful students in the data set.

5.1.1 Evaluation on training data set

In what follows, we present the evaluation metrics for the training data set in Table 3.

	Training data set
Accuracy	0.02
Precision	0.03
Recall	0.01

Table 3. Binary image-classification model evaluation on the training data set.

5.1.2 Evaluation on test data set

In what follows, we present the evaluation metrics for the testing dataset in Table 4 without compression and, in Table 5, with compression.

	Testing data set
Accuracy	0.01
Precision	0.012
Recall	0.013

Table 4. Binary image-classification model evaluation on the testing data set without image compression.

	Testing data set
Accuracy	0.001
Precision	0.0012
Recall	0.0013

Table 5. Model evaluation on the testing data set with image compression.

5.2 Execution times

In what follows we explain the relation of the average execution time and average file size of the images in the data set, in Table 6.

Compute execution time for each image in Github. Report average execution time Vs average file size.

	Average execution time (s)	***Average file size (MB)***
Compression	100.2 s	12.4 MB
Decompression	800.1 s	12.4 MB

Table 6: Execution time of the *(Please write the name of the algorithms, for instance, seam carving & LZ77)* algorithms for different images in the data set.

5.3 Memory consumption

We present memory consumption of the compression and decompression algorithms in Table 7.

	Average memory consumption (MB)	***Average file size (MB)***
Compression	634 MB	3.12 MB
Decompression	9 MB	878.12 MB

Table 7: Average Memory consumption of all the images in the data set for both compression and decompression.

5.3 Compression ratio

We present the average compression ratio of the compression algorithm in Table 8.

	Healthy Cattle	***Sick Cattle***
Average compression ratio	1:23	1:34

Table 8: Rounded Average Compression Ratio of all the images of Healthy Cattle and Sick Cattle.

6. DISCUSSION OF THE RESULTS

Explain the results obtained. Are precision, accuracy and sensibility appropriate for this problem? Is the model overfitting? Is memory consumption and time consumption appropriate? Is compression ratio appropriate? Does compression changes significantly precision on the test data set? *(In this semester, according to the results, can this improve animal-health classification in the context of PLF?)*

6.1 Future work

Answer, what would you like to improve in the future? How would you like to improve your algorithm and its implementation? What about using discrete cosine transform or wavelet compression?

ACKNOWLEDGEMENTS

Identify the kind of acknowledgment you want to write: for a person or for an institution. Consider the following guidelines: 1. Name of teacher is not mentioned because he is an author. 2. You should not mention websites of authors of articles that you have not contacted. 3. You should mention students, teachers from other courses that helped you.

As an example: This research was supported/partially supported by [Name of Foundation, Grant maker, Donor].

We thank for assistance with [particular technique, methodology] to [Name Surname, position, institution name] for comments that greatly improved the manuscript.

REFERENCES

[1] Wikipedia. 2021. Wikipedia: Image scaling. Retrieved from https://en.wikipedia.org/wiki/Image_scaling#Nearest-neighbor_interpolation

[2] Matthew Giassa. 2021. GIASSA: I – Nearest Neighbour Interpolation. Retrieved from https://www.giassa.net/?page_id=207

[3] IMAGE PROCESSING. 2021. IMAGE PROCESSING: Nearest Neighbor Interpolation. Retrieved from https://www.imageeprocessing.com/2017/11/nearest-neighbor-interpolation.html

[4] Dhanesh Budhrani. 2019. Towards Data Science: How data compression works: exploring LZ77. Retrieved from https://towardsdatascience.com/how-data-compression-works-exploring-lz77-3a2c2e06c097

[5] Wikipedia. 2021. Wikipedia: Lempel–Ziv–Storer–Szymanski. Retrieved from https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Storer%E2%80%93Szymanski

[6] Aditya Sharma. 2019. Image Compression using Seam Carving and Clustering. Retrieved from https://adityashrm21.github.io/Image-Compression/

[7] Sandesh Bhusal. 2019. AlgoPods: Burrows Wheeler Transform. Retrieved from https://medium.com/algopods/burrows-wheeler-transform-c743a2c23e0a

[8] Sully Santos. 2014. CONtexto ganadero: 14 enfermedades sin control oficial atacan al ganado en Colombia. Retrieved from https://www.contextoganadero.com/reportaje/14-enfermedades-sin-control-oficial-atacan-al-ganado-en-colombia

[9] Wikipedia. 2021. Wikipedia: Compresión de Burrows-Wheeler. Retrieved from https://es.wikipedia.org/w/index.php?title=Compresi%F3n_de_Burrows-Wheeler&

[10] Asawari Kulkarni, Aparna Junnarkar. 2015. Gray-Scale Image Compression Techniques: A Review . International Journal of Computer Applications, Volume 131.

[11] Nathanael J. Brittain, Mahmoud R. El-Sakka. 2007. Grayscale true two-dimensional dictionary-based image compression. Journal of Visual Communication and Image Representation, Volume 18, Issue 1.

[12] M. Mozammel Hoque Chowdhury, Amina Khatun. 2012. Image Compression Using Discrete Wavelet Transform. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4.

[13] E. Kougianos, S. P. Mohanty, G. Coelho, U. Albalawi and P. Sundaravadivel. 2016. Design of a High-Performance System for Secure Image Communication in the Internet of Things. IEEE Access, vol. 4.

[14] M. H. Asghari and B. Jalali. 2014. Discrete Anamorphic Transform for Image Compression. IEEE Signal Processing Letters, vol. 21, no. 7. pp. 829-833. pp. 1222-1242.
[15] W. Feng, W. Ju, A. Li, W. Bao and J. Zhang. 2019. High-Efficiency Progressive Transmission and Automatic Recognition of Wildlife Monitoring Images With WISNs. IEEE Access, vol. 7, pp. 161412-161423.
[16] T. H. Thai and R. Cogranne. 2019. Estimation of Primary Quantization Steps in Double-Compressed JPEG Images Using a Statistical Model of Discrete Cosine Transform. IEEE Access, vol. 7, pp. 76203-76216.
[17] BBC. 2021. BBC: Pigs can play video games with their snouts, scientists find. Retrieved from https://www.bbc.com/news/technology-56023720?fbclid=IwAR0oKBDNNnMtut-r5k_-0H9nOSp97jC2w12WVYYcWT8tN5mpeJqzm4CJ3s8