IMAGE COMPRESSION ALGORITHMS FOR PROCESS OPTIMIZATION IN LIVESTOCK FARMINGPRECISION

David González Idárraga
Universidad Eafit
Colombia
dgonzalezi@eafit.edu.co

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

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

ABSTRACT

In the development of the Project we decided to use two different algorithms, one for lossy compression and one for lossless compression, for the first one, the chosen algorithm was Singular Value Decomposition (SVD) algorithm and for the second one, we developed the Huffman Algorithm.

After analyzing the results below, we consider these results successful and in accordance with the expected, because, in themselves, they indicate a correct solution to the initial problem, which as it was said, was to make more efficient the process with which these images are compressed and decompressed, both algorithms, although different, generate shorter execution times and lower memory expenses than if they had never been implemented. Also, during this project we compare the complexity of both algorithm and gave conclusion so in the future we will approach better, faster and efficient projects results knowing what to do in order to improve the actual ones.

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

1. INTRODUCTION

The motivation for this project is to allow an easier management of images in an algorithm that identifies healthy and sick animals, by means of CVS type image compression, since lighter and easier to handle files are required due to the low efficiency and capacity of the technology usually used in rural areas (where livestock activities are carried out). Initially, the first algorithm is required to constantly monitor the animals in real time, since it saves time and effort for the person in charge of the animals

1.1 PROBLEM

The problem that is being look at is the development of an algorithm of compression of images to classify animal health in the context of precision livestock farming, to be used by systems with lower specifications and with greater ease. It is a problem that affects not at small, but at large-scale livestock industry, because if solved it could save a lot of time and resources invested in constantly checking the health of animals used in this industry, which has a direct influence on the entire food trade. And after all, an important part of how our society is built.

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.

In this project wanted to make easier the classification of livestock farming, so we wanted to use an algorithm to compress and decompressed the images, to classify them in healthy and unhealthy livestock. Because of that we decided to use “Singular value decomposition” for the algorithm with lossy image compression, because it uses some algebraic procedures that are efficient, and return a good quality imageif the compression is not big. And Huffman coding for the lossless image compression because it the time it spends and the storage it uses is shorter and is more efficient.

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

2.1 Classification of behavior in housed dairy cows using an accelerometer-based activity monitoring system [Vázquez-Diosdado et al. 2015]

- The problem this article is aiming to solve is to both classify biologically important behavior in dairy cows and to detect transition events between lying and standing.

- In this article used an algorithm called Decision-tree algorithm.

- The algorithm is able to classify three types of biological behaviors:

1. Lying (77.42 % sensitivity, 98.63 % precision).
2. Standing (88.00 % sensitivity, 55.00 % precision).
3. Feeding (98.78 % sensitivity, 93.10 % precision).

Transitions were also detected with (96.45 % sensitivity, 87.50 **%** precision).

2.2 Precision Livestock Farming in Swine Welfare: A Review for Swine Practitioners [Benjamin and Yik 2019]

- The problems this article is aiming to solve are group pig welfare challenges: lameness, body condition, prolapse, pig comfort, antagonistic behavior and recognition of illness.

- In this article they talk about an algorithm referred as Hansen's algorithm: it recognized pigs from three regions: the snout and wrinkles above the snout, prevalent marking at the top of the head, and the eye regions.

- The mounted program used digital photos and differentiated 10 pigs (Figure 3) [47], with 96.7% accuracy

2.3 A Combined Offline and Online Algorithm for Real - Time and Long-Term Classification of Sheep Behaviour : Novel Approach for Precision Livestock Farmi ng [Vázquez-Diosdado et al. 2019]

- The problem this article is aiming to solve is "concept drift", which occurs when systems are presented with new or changing conditions, and/or in scenarios where training data is not live sensed.

- In this article used many algorithms: Offline KNN algorithm, Online MeanAMag calculation and Online combined algorithm.

- The combined algorithm shows a 25.90% higher performance when compared to using the offline algorithm only, and a 10.88% higher performance when using the online algorithm only. The average accuracy of the combined algorithm was 85.18%.

2.4 A Combined Offline and Online Algorithm for Real - Time and Long-Term Classification of Sheep Behaviour : Novel Approach for Precision Livestock Farming [Vázquez-Diosdado et al. 2019]

- The problem this article is aiming to solve is "concept drift", which occurs when systems are presented with new or changing conditions, and/or in scenarios where training data is not live sensed.

- In this article used many algorithms: Offline KNN algorithm, Online MeanAMag calculation and Online combined algorithm.

- The combined algorithm shows a 25.90% higher performance when compared to using the offline algorithm only, and a 10.88% higher performance when using the online algorithm only. The average accuracy of the combined algorithm was 85.18%.

.

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.

3.2.1 Image Scaling

Image scaling is a computer graphics process that increases or decreases the size of a digital image.

When scaling a raster graphics image, a new image with a higher or lower number of pixels must be generated.

Image scaling can be interpreted as a form of image resampling or image reconstruction from the view of the Nyquist sampling theorem.

An image size can be changed in several ways:

Nearest-neighbor interpolation:

One of the easy ways of increasing image size is nearest-neighbor interpolation, replacing every pixel with the nearest pixel in the output; his means multiple pixels of the same color will be present.

Bilinear and bicubic algorithms:

This algorithm reduces contrast (sharp edges) in a way that may be undesirable for line art.

Box sampling:

is to consider the target pixel a box on the original image, and sample all pixels inside the box. This ensures that all input pixels contribute to the output. The major weakness of this algorithm is that it is hard to optimize.

Deep convolutional neural networks:

This method uses machine learning for more detailed images such as photographs and complex artwork.

https://en.wikipedia.org/wiki/Image_scaling#Algorithms

3.2.2 Fractal compression:

Fractal compression is a lossy compression method. It seeks to construct an approximation of the original image that is accurate enough to be acceptable.

The method is best suited for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image.

To do fractal compression, the image is divided into sub-blocks. Then for each block, the most similar block if found in a half size version of the image and stored. This is done for each block.

Then during decompression, the opposite is done iteratively to recover the original image.

Basically, the compression process, very broadly speaking, is as follows:

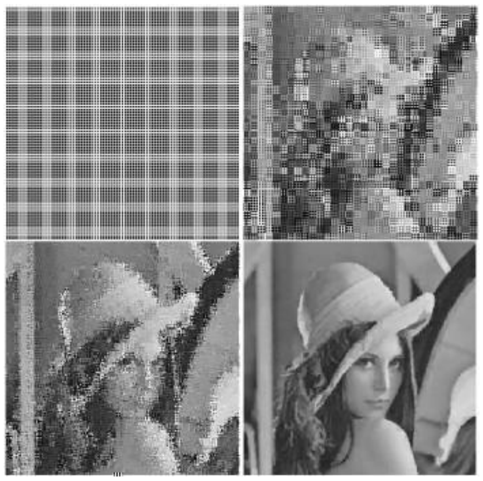
The source image is divided into subsets called domain regions, on which redundancies will be searched within the image.

For each domain region a range region is chosen, larger in size than the domain region.

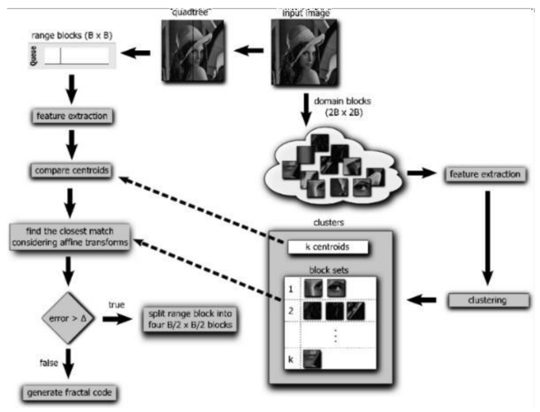
All possible range regions are not rotated, scaled and a symmetry is applied (in short, an affine transformation), choosing the range region that, together with the affine transformation, most closely approximates the domain region.

The choice of the range region, together with the affine transformation, are stored in the fractal file, and will constitute the patterns for decompression and thus reconstruct the original image.

The decompression process is summarized in iterating a sufficient number of times all the affine transformations stored on the range regions until reaching an invariant set, the attractor, which is a good approximation of the original image (as it is a lossy compression , it will never be a pixe l- by-pixel replica of the original).



Decompression process of a fractal image



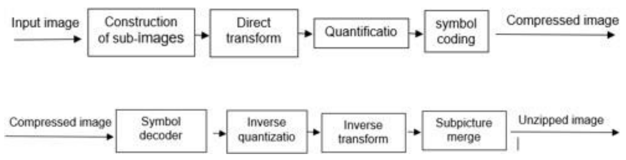
3.2.3 Discrete cosine transform

The discrete cosine transform (DTC) algorithm is the key tool of JPEG compression. It is the most common function that provides spatial compression, capable of detecting the variation of information between an area and contiguous to a digital image.

DCT is an orthogonal transformation method that decomposes an image to its spatial frequency spectrum. It is used a lot in compression tasks. It is a type of Fourier-related Transform, similar to discrete Fourier transforms (DFTs), but only using real numbers.

The image must be divided into blocks of 8x8 pixels to better apply the redundancy of information. The encoding process is performed with the main transform formula. As the image is a two-dimensional signal, the equation must be adapted as a two-dimensional equation.

The DCT coding is based on the following scheme:



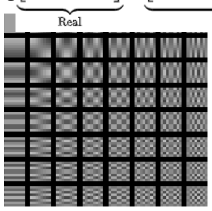
First, the image to be compressed is divided into blocks or sub images of reduced size on which the DTC is applied. The result of the transformation of each block is quantized and then efficient codes are applied to transmit or store this information.

Subsequently, the image has to be recomposed from the blocks into which it was originally divided.

Complexity:
The whole 3-D DCT calculation needs stages, and each stage involves butterflies. The whole 3-D DCT requires butterflies to be computed. Each butterfly requires seven real multiplications (including trivial multiplications) and 24 real additions (including trivial additions). Therefore, the total number of real multiplications needed for the is stage is, and the total number of real additions i.e. including the post-additions (recursive additions) which can be calculated directly after the butterfly stage or after the bit-reverse stage are given by:

Representation of the base images of the cosine transform 2D 8

$$\underbrace{\left[\frac{3}{2}N^3\log_2 N\right]}_{\text{Real}} + \underbrace{\left[\frac{3}{2}N^3\log_2 N - 3N^3 + 3N^2\right]}_{\text{Recursive}} = \left[\frac{9}{2}N^3\log_2 N - 3N^3 + 3N^2\right].$$



3.2.4 Seam carving:

Seam carving is an algorithm for content-aware image resizing. It functions by establishing a number of seams (paths of least importance) in an image and automatically removes seams to reduce image size or inserts seams to extend it.

Most of the original image cropping algorithms will distort the image, and this algorithm proposes an energy-based principle to accommodate cropping. Simply put, each pixel value is assigned an energy value, and then based on this pixel value, 8 connected domains are used for dynamic programming to get the minimum value, and then, applying this row algorithm per row or column per column you will get one power line, which is actually the phase. The line of the pixel with the smallest pixel value of two adjacent rows (columns) is removed from the original image. How many of those lines can be removed depends on the scale to be trimmed. The energy mentioned above is actually the same as the pixel gradient. The very important information in the image, its edge outline texture, etc. it will change enormously, and the gradient will also be obvious. So you may think that such important content has a lot of energy.

The basic steps of the algorithm:

1. Assign an energy value to each pixel

2. Find the eight connected paths of the pixel with the lowest energy value.

3. Remove all pixels in the path

4. Repeat steps 1-3 above until the number of deleted rows / columns reaches the ideal state.

3.3 Lossless Image-compression alternatives

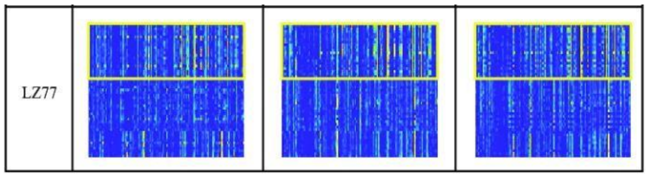
In what follows, we present different algorithms used to compress images

3.3.1 LZ77

This algorithm uses a wave heuristic to scan the image and suffix trie, to represent the dictionary, which is a window in previously encoded pixels.

The complexity of this algorithm is given by N. And the way the LZ77 works is by the dictionary- based scheme, which is designed to take advantage of the correlation between pixels in grayscale images. This wants to represent a block of uncompressed pixels by a pointer to the best approxima te occurrence of that block in the compressed part of the image [1].

In this algorithm occurs a process of matching where there are multiple tasks to evaluate the pixels, according to measures. And after this process, and considering every possible match tooted in the region evaluated, the largest match is considered the best match



https://en.wikipedia.org/wiki/LZ77_and_LZ78

Figure 1

Example of LZ77

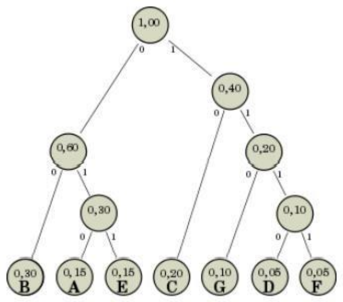
3.3.2 Huffman coding

Huffman is algorithm for data compressing, this algorithm uses a table with certain codes, in which it makes reference to a symbol. The choice of the representation of these symbols isn't random, due to each of this representation creates a prefix code

This algorithm work in $O(n \ log \ n)$ and n represents the number of symbols, however if it's in linear time I, its O(n)

This algorithm works by the construction of a tree and the symbols are in nodes, it is organized by weight and frequency, it continues that process until there is only one node. In the construction of this tree each symbol is assign values of 1 and 0. In case there is only one symbol, 1 is always first and 0 second.

This algorithm is use I other method compression, as multimedia codec deflation such as JPEG and MP3.



https://en.wikipedia.org/wiki/Huffman_coding

Figure 2.

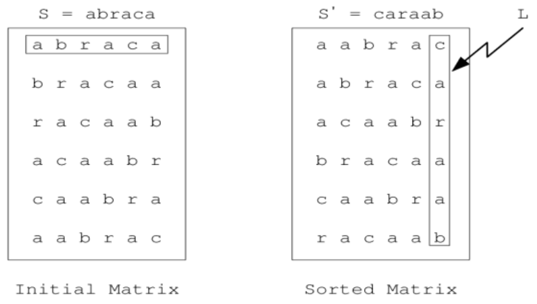
Example of Huffman coding Wikipedia

3.3.3 Burrows-Wheeler transform

Also known as block-sorting compression, in this compression algorithm the value of the characters in the string are kept, due to the transformation of the char string that permutes the order of the characters.

In this compression, all the possible characters entrance rotations are organized column by column, until we got an easy codification. Also, this method is reversible, and we don't need to store any additional data except the position of the first original character.

The complexity in this algorithm is of $O(n)$



https://www.researchgate.net/figure/An-example-of-a-Burrows-Wheeler-transformation_fig3_3225483

Figure 3

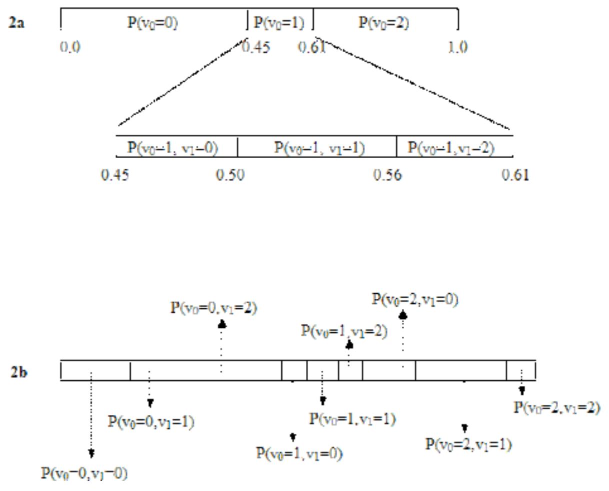
Example Burrows-Wheeler transform

https://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform

3.3.4 ARITHMETIC CODIFICATION

In arithmetic codification the algorithm is given an image with the value of its pixels in a matrix, this algorithm codifies the information of the image with a string of values between 1 and 0 in the line of real numbers, in this way, the image data obtain the lowest size. Even though this technique is easy, is not that efficient at the moment of compressing image data.

This method allows a lossless compression and as in Huffman codification, it creates a string of values between 1 and 0. However to use this method, first you need to define a pattern prediction model, that could be easily found in the message to encode, these models allows that the codification produce an optime codification.



https://en.wikipedia.org/wiki/Arithmetic_coding

Figure 4

Example of Arithmetic coding

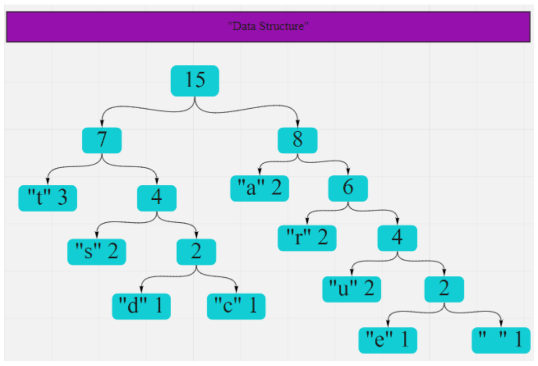
4. ALGORITHM DESIGN AND IMPLEMENTATION

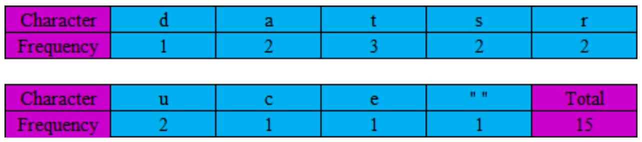
In what follows, we explain the data structures and the algorithms used in this work. The implementations of the data structures and algorithms are available at Github[1].

4.1 Data Structures

The data structure used in the image compression algorithm was the binary tree, specifically implemented in the optimal coding tree (Huffman coding).

The binary tree is a data structure which is composed of root, branch and leaf, in which each node can have one left and one right child (cannot have more than two children). Huffman coding is implemented by constructing a binary tree of nodes from a list of nodes, whose size depends on the number of symbols n. The nodes contain two fields, the symbol and the weight.





Character	d	a	t	s	r
Frequency	1	2	3	2	2

Character	u	c	e	" "	Total
Frequency	2	1	1	1	15

Figure 5. Huffman Tree from the string "Data Structure"

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 used Huffman algorithm and SVD algorithm of compression, one with lossless and the other with lossy compression. But both return images with great quality we also explain how they work.

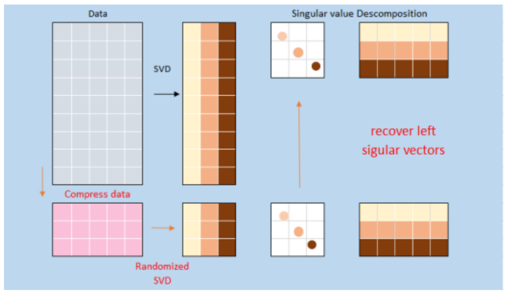


Figure 6. Brief explanation of how SVD algorithm works (Made in Excel by Stefanny Escobar)

[1]https://github.com/StefannyEscobar/ST0245-003/tree/master/proyecto

4.2.1 Lossy image-compression algorithm

The algorithm of that we use is called “Singular Value Decomposition” to understand the algorithm we have to understand the algebra behind this. So, is a matrix factorization method that eigen decomposition of a square matrix (n x n) to any matrix (n x m). We have an original matrix M that we want to decompose, -is left singular matrix (columns are left singular vectors). U columns contain eigenvectors of matrix MM^t. Σ-is a diagonal matrix containing singular (eigen)values, V-is right singular matrix (columns are right singular vectors). V columns contain eigenvectors of matrix M^tM(Hinno-2021).

Now, talking about the algorithm itself, we use some libraries. So, the way of the image is easier to manipulate. The idea is that the images is in greys instead of colors due to in colors it contents mor values than in greys.
Singular value decomposition refactors an image into 3 matrix, the idea is that with the singular values the image is refactor so with a set smaller values we have the new representation of the image, and we reduce the require storage.

We star by reading the image. Converting the integer into doble data type, then calculate the require rank “k",after thatperforms to obtain U,∑ V matrices, then apply approxima- tion on ∑ matrices, regenerate the matrix and remove
singularity, then convert double data type into integers again Finally compressed image is created and display.

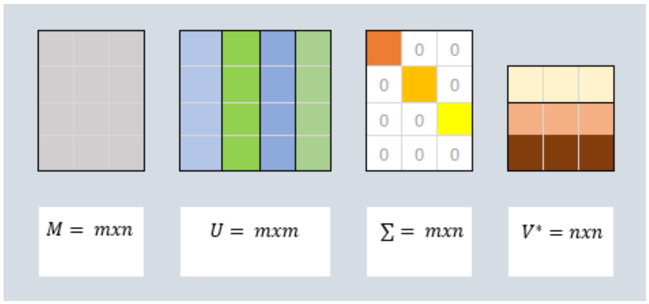


Figure 7. Image using SVD

4.2.2 Lossless image-compression algorithm

Huffman coding is a lossless data compression algorithm, we explained before in 3.3.2, Basically in this algorithm there’sa variable-length code is assigned to input different characters. It fist evaluated the frequency of the characters.

It first have to create a Huffman tree, and another one to traverse the tree to find codes.

4.3 Complexity analysis of the algorithms

Singular Value Descomposition	Time complexity
Compression	O(N)
Decompression	O(N)

***Table* 1:** Time Complexity of the image-compression and image-decompression algorithms N is the width of the matrix and M represents the length of the image matrix

Huffman	Time complexity
Compression	O(N*M)
Decompression	O(N)

***Table* 2:** Time Complexity of the image-compression and image-decompression algorithms N is the width of the matrix and M represents the length of the image matrix

4.4 Design criteria of the algorithm

The algorithms were chosen based in two main criteria: the data structure and the coding method

The lossless compression algorithm Singular Value Decomposition uses relatively simple algebraic procedures it uses relatively simple procedures that can be easily understood, moreover, it is efficient. This algorithm, however, is not so good in terms of compression. Since if we choose a very high compression, the resulting image is of very poor quality.

Huffman's algorithm uses a tree structure to store the pixel values of the in the image according to their probability of occurrence, so that given a grayscale image, the tree extent is reduced, thanks to the high probability of finding many numerous repeated values. Such an assignment of probabilities is given in values of ones and zeros, a factor

Alternatives were explored, such as the LZW algorithm. Which uses a hash table structure, and its compression process can achieve a lower memory complexity than the Huffman However, in terms of time complexity was much more efficient than that achieved with the Huffman algorithm.

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.

5.2 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.

Singular Value Descomposition	*Avarage run time (MB)*	*Average file size (MB)*
Compression	4.3 s	0.138 MB
Decompression	14.7 s	0.132 MB

Table 3: Execution time of SVD for different images in dataset.

Huffman algorithm	*Avarage run time (MB)*	*Average file size (MB)*
Compression	0.7333 s	0.135 MB
Decompression	0.5740599s	0.132 MB

Table 4: Execution time of Huffman Algorithm for different images in data set.

5.3 Compression ratio

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

	Healthy Cattle	*Sick Cattle*
Average compression ratio	2:1	2:1

Table 5: Rounded Average Compression Ratio of all the images of Healthy Cattle and Sick Cattle.

6. DISCUSSION OF THE RESULTS

Two lossy and lossless image compression algorithms the Singular Value Decomposition and the Huffman algorithm. Both, in different ways differently, fulfill the objective of compression and decompression and can be compared in terms of their time efficiency, their time consumption, their memory usage, and the compression ratio provided by each algorithm

6.1 Future work

Thinking about future developments, we would mainly like to continue doing trials and tests with our algorithms, to improve them to their most efficient forms and to their maximum potential, even if this means revisiting them several times, and not only this, but we are also interested in the development of other algorithms, in order to test if they can give better results than those currently obtained, because it never hurts to analyze alternative solution paths. We are not reluctant to try solutions such as discrete cosine transform or wavelet compression, primarily cause this can generate a greater learning and open us to ideas and paths that maybe we did not see at our last stage of work.

ACKNOWLEDGEMENTS

The author is supported by Sapiencia, grant funded by the municipality of Medellín, specifically with the benefit called “Fondo EPM”. I give a big thank you to this foundation as this would have not been possible without their help and contribution. I would like to thank the “Vicerrectoría de Descubrimiento y Creación, Universidad EAFIT”, for their support in this research and development, finally, but not less important, I would like to thank Mauricio Toro, my teacher in this class and constant company at the project, as for the same thing, a last acknowledgment to Simon Marin. This work is what it is today thanks to perseverance and constant interest.

REFERENCES

[1] Wikipedia. 2017. WikipediA: the Free Encyclopedia. Retrieved from https://www.wikipedia.org/.

[2]. Vázquez-Diosdado, J. et al., 2015. Classification of behaviour in housed dairy cows using an accelerometer- based activity monitoring system, Animal Biotelemetry.

[3].Vázquez-Diosdado, J., Paul, V., Ellis, K., Coates, D., Loomba, R. and Kaler, J., 2019. A Combined Offline and Online Algorithm for Real-Time and Long-Term Classification of Sheep Behaviour: Novel Approach for Precision Livestock Farming, Sutton Bonington: MDPI.

[4]. Reference: Benjamin, M. and Yik, S., 2019. Precision Livestock Farming in Swine Welfare: A Review for Swine Practitioners, East Lansing: MDPI.

[5]. Herinaina Andriamandroso, A. et al., 2017. Development of an open-source algorithm based on inertial measurement units (IMU) of a smartphone to detect cattle grass intake and ruminating behaviors, ScienceDirect.

[6]. Vincent Tabora. 2019 JPEG Image Scaling Algorithms(September 2019). Retrieved august 8, 2021 from

https://users.cs.northwestern.edu/~agupta/_projects/image_processing/web/FractalImageCompression/

[7] Yaciro Cabezas, Jairo Guevara, Algoritmo de compresión y reconstrucción de imagen aplicando la teoría de wavelets (2005) Retrieved august 8, 2021 from http://repositorio.unicauca.edu.co:8080/bitstream/handle/1 23456789/2136/ALGORITMO%20DE%20COMPRESIÓN %20Y%20RECONSTRUCCIÓN%20DE%20IMÁGENES%20FIJAS.pdf?sequence=1&isAllowed=y

[8] Jonathan Blow Arithmetic Coding, Part 2 (September 2003) Retrieved august 8, 2021 from

http://number-one.com/product/Arithmetic%20Coding,%20Part%20 2/ind ex.html

https://zephyrnet.com/es/algoritmo-de-tallado-de-costuras- una-forma-aparentemente-imposible-de-cambiar-el- tamaño-de-una-imagen/

https://programmerclick.com/article/4510360417/

[9] Risto Hinno Simple SVD algorithms (January 31 2021)retrieved September 27,2021 from https://towardsdatascience.com/simple-svd-algorithms- 13291ad2eef2

[10] H R Swathi Image compression using singular value

Decomposition (2017) retrieved (September 28. 2021)from https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042082/pdf

[11] Karthikeya Boyini Huffman Coding Algorithm (July 6 2018) retrieved September 28. 2021 from https://www.tutorialspoint.com/Huffman-Coding-Algoritm