Image compression in the context of Precision Livestock Farming

Analysis of the use of image compression in the context of animal health

Sebastián Amaya
Mathematical Sciences
Universidad Eafit
Medellín Antioquia
Colombia
saamayac@eafit.edu.co

Daniel Gómez Vargas
School of Engineering
Universidad Eafit
Medellín Antioquia
Colombia
Dgomezv9@eafit.edu.co

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

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

ABSTRACT

The text revolves around image compression in the context of precision livestock farming (PLF), it starts by analyzing different approaches to PLF and at the same time explain some of the algorithms used in image compression to give a clearer picture of the subject at hand, then one algorithm is chosen, to be explored in depth.

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

1. INTRODUCTION

The world population is expected to peak at the beginning of the 22nd century with an expected total population of 10.88 billion people [6], this presents a lot of challenges, one of them being is food supply or food production, in that same note both production and demand for animal-derived products have steadily grown through the years [5],

Products	Production (1961 - 2018)	Per capita supply (1961 - 2013)
Meat	+271.06 Million Tonnes	+20.14 kg
Milk	+497.66 million Tonnes	+14.45 kg
eggs	+67.75 million Tonnes	+4.64 kg

this growth combined with the climate crisis, has raised the need for farms to be more efficient and at the same time prioritize the reduction of the disruption in the environment, as well as animal wellbeing, which are all objectives of precision livestock farming (PLF) [4], PLF is "...a holistic approach that adds information and communication technologies (ICT) to improve the farming process". one specific part of PLF is the use of image recognition, mainly for identification, checking animal health and so forth, image recognition in PLF can generate a lot of data, and over time this can become problematic because of the amount of storage needed, as well as other factors, one solution is image compression, which can go a long way in helping saving energy, and hardware capacity.

1.1. Problem

The problem of finding an algorithm with the best efficiency and better image compression to identify very accurately when an animal is sick or healthy in the context of livestock farming, would be the result of automation and optimization of the work to generate order, clear and accurate information, which correctly recognize which are the animals that present pictures of risky health and thus reduce the risk of diseases, viruses, and infectious batteries and allow a rapid intervention.

1.3 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.

For the solution of the project we use the image compression with lossy DCT (Discrete Cosine Transform) algorithm that will help us in the process by performing a compression and decompression of images with very little data loss and a very high quality creating in this way, a fast and efficient algorithm at the time of execution.

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 Automatic weight estimation of individual pigs using image analysis

In this work they tested out the possibility of measuring pigs weight using image analysis, this study contains measurements on four pens of grower pigs (each of the pens monitor by a top view camera), shape recognition techniques were employed to identify each pig (this by using the shapes painted on their back),

Figure 1: “frame of a video showing a top view of one of the four pig pens in the research barn.” [1]

after the system identifies a pig, the weight estimation process begins; first, the pig is located via an ellipse fitting algorithm, then the area the pig occupies inside the ellipse is calculated, and lastly, the weight is estimated by a dynamic model. This process achieved an accuracy of 97.5% at a group level, better than existing automated tools (95%) or walk-through systems (97%). [1]

2.2 Lameness detection of dairy cows based on the YOLOv3 deep learning algorithm and a relative step size characteristic vector

This work proposes a solution to the problem of lameness (“the clinical manifestation of painful disorders, mainly related to the locomotor system, resulting in impaired movement or deviation from normal gait or posture”[3]) in dairy cows and by extension, the cow’s welfare and productivity, [3] in this study a method base on the YOLOv3 deep learning algorithm and relative step size characteristic vectors is used, the method first calculates the relative step size of the cow's legs (front and rear) using the leg’s coordinates, then the relative step size characteristic vectors is constructed and lastly, a trained Long Short-Term Memory (LSTM) classification model classified the cows in lameness and non-lameness based on the characteristic vector.

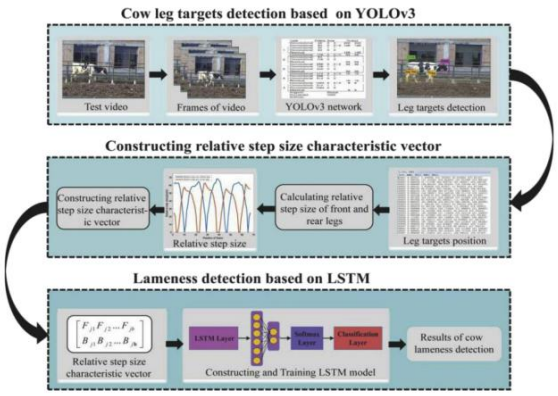
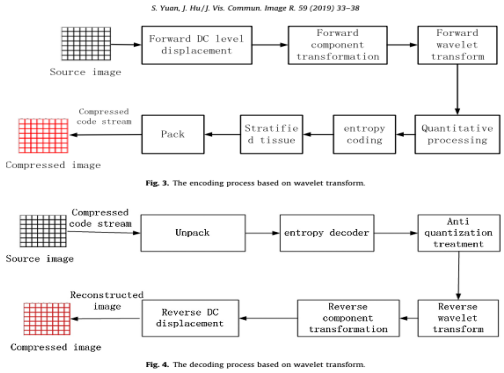


Figure 2: “Technical route of the proposed method” [2]

This method resulted in a higher accuracy rate as well as a higher True positive rate compared to other methods like the support vectors machine or decision tree classifier. [2]

2.3 Research on image compression technology based on Huffman coding



2.4 ESTUDIO DE LA CODIFICACIÓN PREDICTIVA DE IMÁGENES BASADA EN CÓDIGOS RICE

In this work, they set two main objectives. The first objective is to achieve a minimum sending bit rate by predictively coding images with Rice codes, and the second is to see if the bit rate differs from other coding techniques.

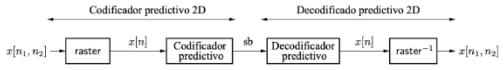


Figure 3: Schematic of a predictive coding system for images.

they found that it is a good option to use an adaptive predictor and adaptive Rice coding, as long as our priority is the minimum bit rate value, and the complexity of the algorithm is not relevant because when using the JPEG-LS predictor it was observed that the bit rate is lower than using a fixed predictor, similarly, with an adaptive predictor the bit rate is reduced, but the implementation of this is a little more complex.

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 discrete cosine transform compression

The process starts by separating the image in blocks of eight by eight pixels which are then normalized, so they range from –128 to 127 instead of 0 to 255, then each of the 8 x 8 blocks (this algorithm works with 8x8 blocks of an image) is separately encoded with a DCT coefficient (each of the coefficients represents a weight of the cosine wave corresponding to the location), the output is the coefficient matrix, the complexity of this algorithm is O(n^4) (for a specific algorithm) for each 8x8 block [9]

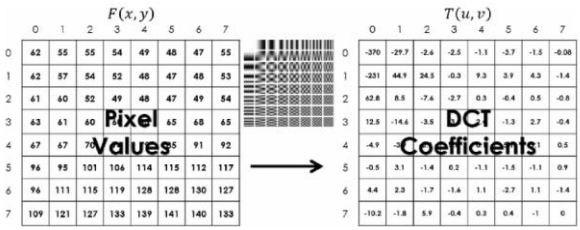


Figure 4: Representation of an 8x8 pixel table converted into a DCT Coefficient table

3.2.2 Quantization compression

Quantization compression is a lossy compression algorithm it works by a process of eliminating higher frequency data (it is applied after the DCT in JPEG image compression), the quantization function is then used to change the matrix (where every coefficient is divided by a corresponding quantization value) (the matrix of quantization values is normally called quantized block), as previously stated the goal of this algorithm is to reduce most of the higher frequency data, the complexity of the O(n^2) or O(N) n being every pixel in the image.[9]

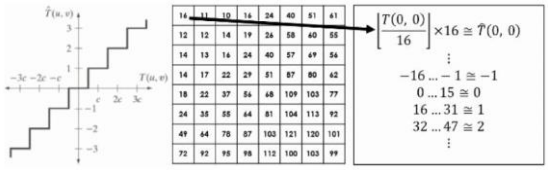


Figure 5: Representation of Quantization compression

3.2.3 Fractal compression

It is a Lossy Image-compression algorithm also used in image graphics representation, this algorithm represented the image in fractal code, and this fractal code is then used to decode it. [10]

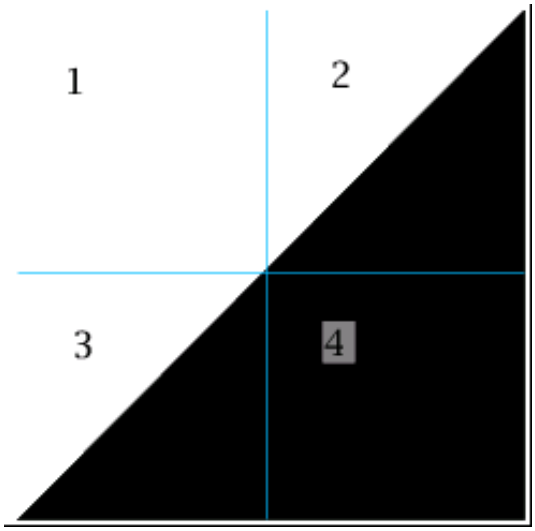


Figure 6: "triangles; an example to show how fractal compression works" [11]

3.2.4 Color cell compression

The Color cell compression is a Lossy Image-compression algorithm used for color images, it is based on block truncation coding, and it has three phases and can achieve a compression of 2 bits/pixel.[12]

(182, 121, 242)	(142, 129, 138)	(230, 9, 162)	(190, 17, 58)
(22, 153, 82)	(238, 161, 234)	(70, 41, 2)	(30, 49, 154)
(118, 185, 178)	(78, 193, 74)	(166, 73, 98)	(126, 81, 250)
(214, 217, 18)	(174, 225, 170)	(6, 105, 194)	(222, 113, 90)

152	133	92	73
105	192	45	54
164	145	103	113
194	203	85	143

1	1	0	0
0	1	0	0
1	1	0	0
1	1	0	1

Figure 7: On the right a 4x4 color cell E, on center luminance channel of example cell E, on the left Bitmap B of example cell E [12]

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 Huffman Compression

In Huffman coding, input string characters are assigned with variable length codes (bit sequences), and the frequency of individual

characters determines the allocated code length.

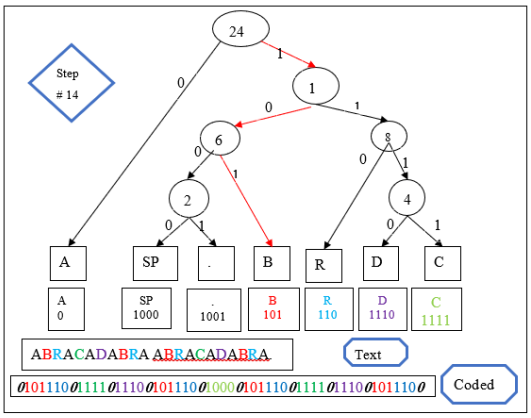


Figure 8: Representation of Huffman compression

3.3.2 LOCO-I lossless image compression

the main objective driving the design of LOCO-I is to systematically “project” the image modeling into a low complexity plane, both from a modeling and coding perspective.

```
B = B + ε;     /* accumulate prediction residual */
N = N + 1;     /* update occurrence counter */
/* update correction value and shift statistics */
if ( B ≤ −N ) {
   C = C − 1; B = B + N;
   if ( B ≤ −N ) B = −N + 1;
}
else if ( B > 0 ) {
   C = C + 1; B = B − N;
   if ( B > 0 ) B = 0;
}
```

Figure 3: Bias computation procedure

Figure 9:

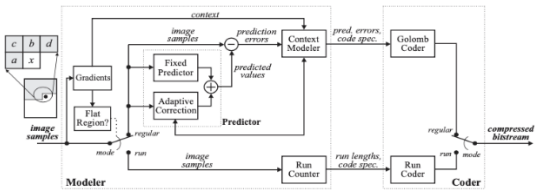


Figure 10:

3.3.3 Rice coding

In this work, monochromatic images of 8 bits per pixel were considered. Therefore, each pixel takes an integer value between 0 and 255 (X = {0,1, ...,255}).

Two predictors were considered: a fixed predictor and an adaptive predictor. The fixed predictor used is the first difference predictor. In the fixed predictor, each sample x[n] resulting from the raster scan is predicted by a first-order predictor with a1 = 1:

$$\tilde{x}[n] = x[n-1]$$

$$\begin{aligned}
\mathbb{E}[\text{compression ratio}] &= \sum_{k=1}^{\infty}(\text{compression ratio of } k\text{−run}) \cdot \mathbb{P}[\text{bit is part of } k\text{−run}] \\
&= \sum_{k=1}^{\infty} \frac{b+1+\lfloor 2^{-b}(k-1) \rfloor}{k} \cdot kp^{k-1}(1-p)^2 \\
&= (1-p)^2 \sum_{j=0}^{\infty}(b+1+j) \cdot \sum_{i=j2^b+1}^{(j+1)2^b} p^{i-1} \\
&= (1-p)^2 \sum_{j=0}^{\infty}(b+1+j) \cdot \left(p^{2^b j} - p^{2^b(j+1)}\right) \\
&= (1-p) \cdot \left(b + \sum_{m=0}^{\infty} p^{2^b m}\right) \\
&= (1-p) \cdot \left(b + \left(1-p^{2^b}\right)^{-1}\right)
\end{aligned}$$

figure 11:

3.3.4 A Block-sorting Lossless Data Compression Algorithm

This algorithm transforms a string S of N characters by forming the N rotations (cyclic shifts) of S, sorting them lexicographically, and extracting the last character of each of the rotations. A string L is formed from these characters, where the ith character of L is the last character of the ith sorted rotation

File	Size (bytes)	CPU time/s compress	CPU time/s decompress	Compressed size (bytes)	bits/ char
bib	111261	1.6	0.3	28750	2.07
book1	768771	14.4	2.5	238989	2.49
book2	610856	10.9	1.8	162612	2.13
geo	102400	1.9	0.6	56974	4.45
news	377109	6.5	1.2	122175	2.59
obj1	21504	0.4	0.1	10694	3.98
obj2	246814	4.1	0.8	81337	2.64
paper1	53161	0.7	0.1	16965	2.55
paper2	82199	1.1	0.2	25832	2.51
pic	513216	5.4	1.2	53562	0.83
progc	39611	0.6	0.1	12786	2.58
progl	71646	1.1	0.2	16131	1.80
progp	49379	0.8	0.1	11043	1.79
trans	93695	1.6	0.2	18383	1.57
Total	3141622	51.1	9.4	856233	-

Table 1: Results of compressing fourteen files of the Calgary Compression Corpus.

Figure 12:

final char (L)	sorted rotations
a	n to decompress. It achieves compression
o	n to perform only comparisons to a depth
o	n transformation} This section describes
o	n transformation} We use the example and
o	n treats the right-hand side as the most
a	n tree for each 16 kbyte input block, enc
a	n tree in the output stream, then encodes
i	n turn, set $L[i]$ to be the
i	n turn, set $R[i]$ to the
o	n unusual data. Like the algorithm of Man
a	n use a single set of probabilities table
e	n using the positions of the suffixes in
i	n value at a given point in the vector $R
e	n we present modifications that improve t
e	n when the block size is quite large. Ho
i	n which codes that have not been seen in
i	n with ch appear in the {\em same order
i	n with ch. In our exam
o	n with Huffman or arithmetic coding. Bri
o	n with figures given by Bell~\cite{bell}.

Figure 1: Example of sorted rotations. Twenty consecutive rotations from the sorted list of rotations of a version of this paper are shown, together with the final character of each rotation.

Figure 13:

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 structure

In this project, we focused especially on data matrices because it allowed us to perform the DCT (Discrete Cosine Transform) lossy compression code, this made it easier for us to execute tables of quantification, compression and decompression of images with little loss of quality in the process.

```
[[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16]
 [  2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32]
 [  3   6   9  12  15  18  21  24  27  30  33  36  39  42  45  48]
 [  4   8  12  16  20  24  28  32  36  40  44  48  52  56  60  64]
 [  5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80]
 [  6  12  18  24  30  36  42  48  54  60  66  72  78  84  90  96]
 [  7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112]
 [  8  16  24  32  40  48  56  64  72  80  88  96 104 112 120 128]
 [  9  18  27  36  45  54  63  72  81  90  99 108 117 126 135 144]
 [ 10  20  30  40  50  60  70  80  90 100 110 120 130 140 150 160]
 [ 11  22  33  44  55  66  77  88  99 110 121 132 143 154 165 176]
 [ 12  24  36  48  60  72  84  96 108 120 132 144 156 168 180 192]
 [ 13  26  39  52  65  78  91 104 117 130 143 156 169 182 195 208]
 [ 14  28  42  56  70  84  98 112 126 140 154 168 182 196 210 224]
 [ 15  30  45  60  75  90 105 120 135 150 165 180 195 210 225 240]
 [ 16  32  48  64  80  96 112 128 144 160 176 192 208 224 240 256]]
```

```
[[ -3   0   3   4   2   3   6  10   5   8  12  13  11  12  15  19]
 [  0   3   7   8   7   8  12  16  15  18  22  23  23  24  28  31]
 [  3   7  12  14  14  16  20  24  27  31  36  38  38  40  45  49]
 [  4   9  14  17  19  22  27  32  36  41  47  50  51  54  60  64]
 [  4   9  15  20  22  27  33  38  43  48  55  59  62  66  73  78]
 [  5  10  18  23  27  33  40  46  52  58  66  71  75  81  88  94]
 [  8  14  22  29  34  40  48  54  65  71  79  86  91  97 105 111]
 [ 12  18  26  33  39  46  54  61  75  81  89  96 102 109 117 124]
 [  5  13  24  34  44  54  65  73  78  86  97 107 117 127 138 146]
 [  9  16  28  39  49  59  71  79  88  96 107 118 128 139 150 158]
 [ 12  20  33  44  55  67  79  87 101 109 121 133 143 155 167 176]
 [ 13  22  35  48  60  73  86  95 110 119 132 144 156 169 182 191]
 [ 13  22  36  50  64  78  92 101 117 126 140 154 167 182 196 205]
 [ 14  24  39  54  69  84  99 109 126 136 150 166 180 196 211 220]
 [ 18  28  43  60  75  91 107 117 138 148 164 180 196 212 228 238]
 [ 21  31  47  64  80  97 113 123 148 158 174 191 207 224 240 250]]
```

Figure 14: Top: Original matrix from CSV image. Bottom: Matrix after DCT compression algorithm.

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.

4.2.1 Lossy image-compression algorithm

The code we did for the solution of the project is the DCT (Discrete Cosine Transform) This code is mainly used for the well-known .JPEG image format.

DCT is an algorithm based on the cosine function, we take data from the image and get a wavelet sum of the cosine function

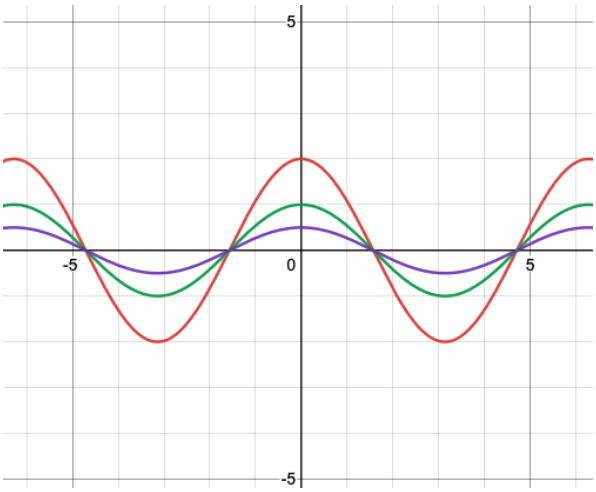


Figure 15: cosine function representation

With this idea, we create 8x8 pixel arrays of the images to perform our DCT algorithm and obtain a compression with minimum data loss.

[1] http://www.github.com/ ????????? /proyecto/

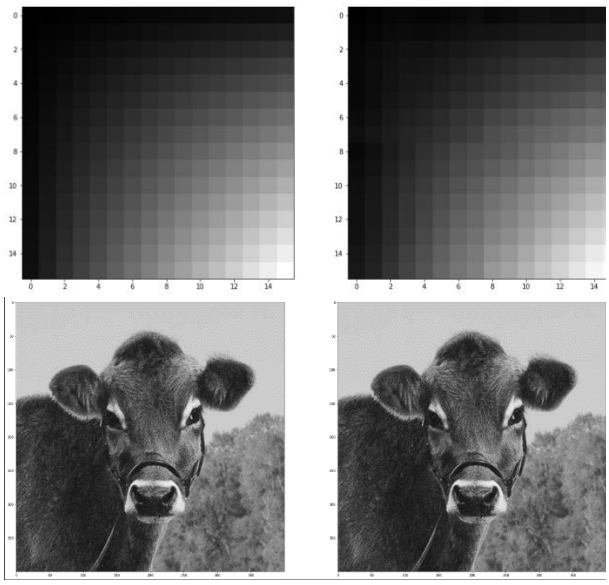


Figure 16-17: comparison between original .CVS image (Left images) and final process after DCT lossy compression algorithm (Right images)

REFERENCES:

[1] Mohammadamin Kashiha, Claudia Bahr, Sanne Ott, Christel P.H. Moons, Theo A. Niewold, Frank O. Ödberg, Daniel Berckmans. Automatic weight estimation of individual pigs using image analysis. Computers and Electronics in Agriculture volume 107, September 2014, Pages 38-44.

[2] Dihua Wu, Qian Wu, Xuqiang Yin, Bo Jiang, Han Wang, Dongjian He, Huaibo Song. Lameness detection of dairy cows based on the YOLOv3 deep learning algorithm and a relative step size characteristic vector. Biosystems Engineering Volume 189, January 2020, Pages 150-163

[3] Annelies Van Nuffel, Ingrid Zwertvaegher, Liesbet Pluym, Stephanie Van Weyenberg, Vivi M. Thorup, Matti Pastell, Bart Sonck and Wouter Saeys. Lameness Detection in Dairy Cows: Part 1. How to Distinguish between Non-Lame and Lame Cows Based on Differences in Locomotion or Behavior. Published online 2015 Aug 28, Retrieve from Lameness Detection in Dairy Cows: Part 1. How to Distinguish between Non-Lame and Lame Cows Based on Differences in Locomotion or Behavior (nih.gov)

[4] Rodrigo García, Jose Aguilar, Mauricio Toro, Angel Pinto, Paul Rodríguez. A systematic literature review on the use of machine learning in precision livestock farming. Computers and Electronics in Agriculture, Volume 179, December 2020

[5] Hannah Ritchie and Max Roser (2017) - "Meat and Dairy Production". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/meat-production' [Online Resource]

[6] Max Roser (2013) - "Future Population Growth". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/future-population-growth' [Online Resource]

[7] M. J. Weinberger, G. Seroussi and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," in IEEE Transactions on Image Processing, vol. 9, no. 8, pp. 1309-1324, Aug. 2000, DOI: 10.1109/83.855427.

[8] Ankur Gupta, Muskan Garg, Apurv Verma, Dushyant Kaushik, Implementing lossless compression during image processing by integrated approach, Materials Today: Proceedings, 2020, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2020.10.052.(https://www.sciencedirect.com/science/article/pii/S2214785320376197)

[9] P. T. Chiou, Y. Sun and G. S. Young, "A complexity analysis of the JPEG image compression algorithm," 2017 9th Computer Science and Electronic Engineering (CEEC), 2017, pp. 65-70

[10] N. M. G. Al-Saidi and A. H. Ali, "Towards enhancing of fractal image compression performance via block complexity," 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT), 2017, pp. 246-251

[11] Marek Drwota, 2 triangles; example to show how fractal compression works, Wikipedia Commons, 2006. Retrieve from:

https://en.wikipedia.org/wiki/Fractal_compression#/media/File:Zasada_dzialania_ifs_1.png

[12] Pins M. (1991) Extensions of the Color-Cell-Compression-Algorithm. In: Thalmann N.M., Thalmann D. (eds) Computer Animation ’91. Springer, Tokyo. https://doi.org/10.1007/978-4-431-66890-9_17

[13] Shuyun Yuan, Jianbo Hu, Research on image compression technology based on Huffman coding, Journal of Visual Communication and Image Representation, Volume 59, 2019, Pages 33-38, ISSN 1047-3203, https://doi.org/10.1016/j.jvcir.2018.12.043.(https://www.sciencedirect.com/science/article/pii/S1047320318303717)

[14] Manglano Bermejo, M. (2020). Codificación predictiva de imágenes usando códigos Rice (Doctoral dissertation).

[15] Abu-taieh, Evon. (2018). The Pillars of Lossless Compression Algorithms a Road Map and Genealogy Tree. International Journal of Applied Engineering Research. 13.

[16] J. Ding, H. Chen, and W. Wei, "Adaptive Golomb Code for Joint Geometrically Distributed Data and Its Application in Image Coding," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 4, pp. 661-670, April 2013, DOI: 10.1109/TCSVT.2012.2211952.