PRECISION LIVESTOCK FARMING: OPTIMIZATION OF BATTERY CONSUMPTION BASED ON COMPRESSION ALGORITHMS.

Sofía Castaño
Universidad Eafit
Colombia
scastanop@eafit.edu.co

Andrés Camilo Álvarez
Universidad Eafit
Colombia
acalvarezv@eafit.edu.co

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

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

ABSTRACT

The agricultural sector is the main source of both financial gains in rural areas and of obtaining animal and plant resources. That is why it is of vital importance that on farms there are methods and techniques to optimize the work within these spaces to ensure a benefit not only for the producer but also for the consumer. In this report we will focus on Precision Livestock Farming (PLF), which describes those sustainably techniques used to improve production, control and profitability through different types of monitoring that allow us to manage better decision making and maintain the care and productivity of each of the animals involved in that process.[1]

Keywords

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

1. INTRODUCTION

As the population increases and our consumption of resources, techniques must be implemented to help us cope with this situation and the challenges that accompany it in a sustainable manner. The PLF since its inception in countries such as the UK, Germany, Denmark, among others, focused on the use of information and communication technologies to reduce investment costs and increase both production and animal welfare.[2]

Currently the PLF is in a challenging stage, since the progress achieved with respect to the power of the computer processors shows the challenge of the correct implementation of these systems, a good management and interpretation of the data obtained, based on sustainable alternatives that benefit both environmental impact and resource consumption and therefore production costs.

1.1. Problem

We are challenged to design an algorithm capable of accurately compressing and decompressing images to reduce and optimize battery consumption during the implementation of a PLF-based monitoring and control method. The purpose of the project is to evaluate these compressed images using an animal health classification algorithm, achieving the highest compression rate possible without affecting the accuracy of the classification by more than 5%.

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.

1.3 Article structure

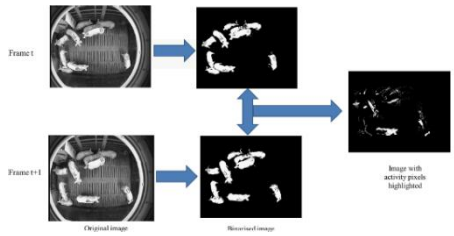
In what follows, in Section 2, we present related work to the problem. Later, in Section 3, we present the data sets and methods used in this research. In Section 4, we present the algorithm design. After, in Section 5, we present the results. Finally, in Section 6, we discuss the results and we propose 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 monitoring of pig locomotion using image analysis.

In this work they monitor individual pig's locomotion in a group housed environment through automated quantify-cation of locomotion levels under using continuous image analysis, which allows to know and predict the behavior of pigs using an ellipse fitting model. First you need to use adaptive histogram equalization to remove the light effects captured by the camera, then each image is binarized to remove the background and finally, each image is segmented in order to find the location of the pigs. To segment the



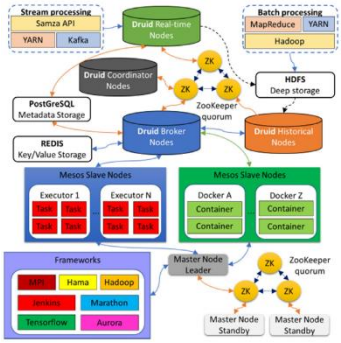
image, the pigs' bodies are extracted as ellipses within each pen.[3]

Illustration 1. Locomotion calculation in eYeNamic tool.

2.2 Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors.

For this problem the sensors of the iPhones were used to monitor the behavior within the livestock, as the iPhone 5SE, 6S, 7S and 8S are equipped with a new IMU, which allowed

to obtain the data of the behavior of the cows. However, having to collect a massive amount of data, these could only be compressed by 43.5%, so they devised a way to compress individual values in a more efficient way, developing a data storage architecture is able to collect data at high frequency and is adaptable easily to many cases. The main originality of the architecture lies on its ability to share the data and applications created by the different teams of scientists from a common database. This architecture is also able to integrate



complementary data such UAV images, and external data from other cloud platforms such as health and production data.[4]

Illustration 2. Components of the proposed architecture

3.3 IoT based Animal Health Monitoring with Naïve Bayes Classification.

The work aims to help and benefit both animal health and farmers using the Wireless Sensor Network and IoT applications. For this purpose, sensors are installed in the farm, which collect the physical information collected such as temperature, heart rate and load cell of the animals. This data is transmitted individually to a micro-controller and then imported into the API that uses HTTP internet connection via GPRS, WIFI or cable connection, and is responsible for the integration and management of sensor data stored in a database. Then, using the Naive Bayes sorting method, the new sensor readings are compared with those previously recorded and in case of a drastic change the user is notified on the PC.[5]

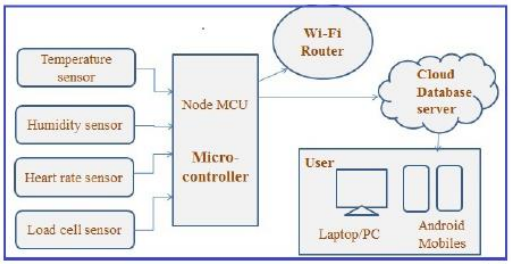
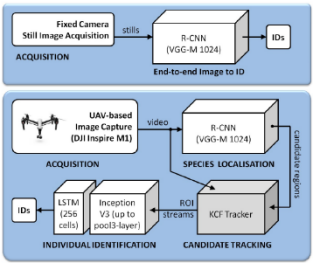


Illustration 3. Block diagram of animal health monitoring

3.4 Visual Localization and Individual Identification of Holstein Friesian Cattle via Deep Learning.

Usually, individual identification of animals in livestock can become intrusive, this is why in this work it was possible to make this identification individually using standard deep learning pipes focusing on the Holstein Friesian race. For identification in particular, it was shown that convolution-based architectures are well suited to learning and distinguishing the properties of the unique dorsal pattern and the structure exhibited by the species individually. The deep network utilized to address this problem is the R-CNN adaptation of the VGG CNN M 1024 network published as part of several other network architecture proposals. The core architecture consists of 5 stacked convolutional layers, which are shared with the region proposal network. Instead of training from scratch, weights were initialized with a model trained on the ImageNet database supplied with the Faster-RCNN Python implementation. The object detection and localization task produces image Regions of Interest (RoIs) or proposals in the form of bounding boxes. It is important to note that this process can take place in a non-intrusive



manner in practically relevant environments, in contrast to most existing identification frameworks that revolve around physical labelling.[6]

Illustration 4. Proposed Component Pipelines.

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-compcression alternatives

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

3.2.1 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. The process to transform the image is: 1. It start reading an image. 2. It calculates the weight/density/energy of each pixel; this is done by an algorithm called gradient magnitude. 3. From the energy, make a list of seams. Seams are ranked by energy, with low energy seams being of least importance to the content of the image. 4. It removes low-energy seams as needed 5. It returns the image transformed.[7]

Illustration 5. Seam Carving Process

3.2.2 Discrete Cosine Transform.

The Discrete Cosine Transform (DCT) works by separating images into parts of differing frequencies. The first step is to break the image into 8x8 block for pixel, then the DCT is applied to each block working from left to right and top to bottom. The next step is called quantization, in which the compression actually occurs, the less important frequencies are discarded, then, only the most important frequencies that remain are used retrieve the image in the decompression process. In the next step the array of compressed blocks that constitute the image is stored in drastically reduced amount of space, and finally, the image is reconstructed through decompression, a process that uses de Inverse Discrete Cosine Transform (IDCT).[8]

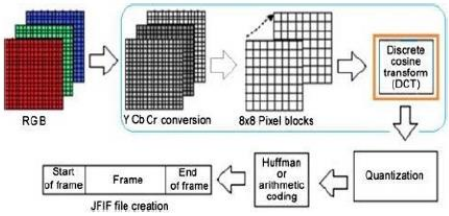


Illustration 6. Discrete Cosine Transform

3.2.3 Image Scaling.

The size of an image can be changed in different ways. Resizing or resampling an image. There are several ways to change an imagen, these are the most common. 1. Nearest Neighbor Scaling: This is the fastest and simplest to implement but doesn't have the best result. 2. Bilinear interpolation: This interpolates pixels much better than Nearest Neighbor. It uses the same approach but the computation is much more complex. 3. Bicubic Interpolation: This is a much slower algorithm, but the results are much smoother looking images with less artifacts. This algorithm produces the best results out of the three discussed.[9]

Illustration 7. Image Scaling Process

3.2.4 Fractal Compression.

Fractal compression is a lossy compression method for digital images, based on fractals. 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. Fractal algorithms convert these parts into mathematical data called "fractal codes" which are used to recreate the encoded image.[10]

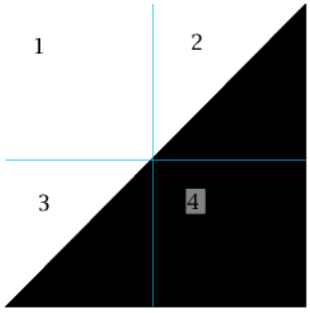


Illustration 8. Fractal Compression Method

3.3 Lossless Image-compression alternatives

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

3.3.1 LZ77.

The LZ77 Compression Algorithm is used to analyze input data and determine how to reduce the size of that input data by replacing redundant information with metadata (XML-formatted data that defines the characteristics of an update). The encoding algorithm is used to take that combination of data and metadata and serialize it into a stream of bytes that can later be decoded and decompressed.

Using the Compression Algorithm: 1. Set the coding position to the beginning of the input stream. 2. Find the longest match in the window for the lookahead buffer. 3. If a match is found, output the pointer P. Move the coding position (and the window) L bytes forward. 4. If a match is not found, output a null pointer and the first byte in the lookahead buffer. Move the coding position (and the window) one byte forward. 5. If the lookahead buffer is not empty, return to step 2. [11]

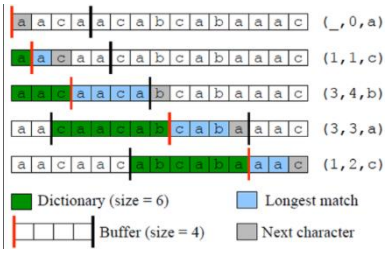


Illustration 9. LZ77 Algorithm

3.3.2 Huffman Coding.

The Huffman algorithm used to compress data is based on binary trees. Mostly used to compress files where the frequency of appearance of each symbol is known. The main idea of the algorithm is to calculate the appearances of each symbol and based on this generate its respective representation in binaries.

Main steps:

1.Create a Huffman tree based on the read file. 2. Go through the tree and assign the corresponding codes. 3. Save the symbol and its code in a table. [12]

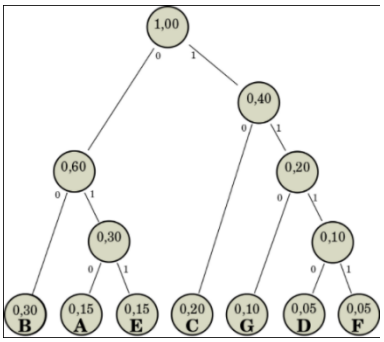


Illustration 10. Huffman Coding Algorithm

3.3.3 LZW.

The Idea relies on reoccurring patterns to save data space. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility. LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. How it works? When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes.

As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table.[13]

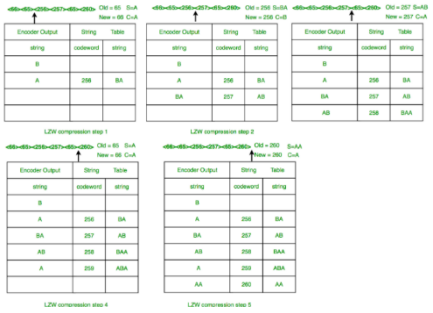


Illustration 11. LZW Algorithm

3.3.4 Wavelets.

Is the capability to send the image or set of images over low-cost telephone lines in a few seconds rather than tens of minutes to an hour or more if compression is not used. With wavelet compression, on-line medical collaboration can be accomplished almost instantaneously via dial-up telephone circuits. By modeling the linear, compressed, smooth coefficients directly and then using the inverse transform on

the model saves computation time and space but also improves the quality of the model by eliminating noise. Once the model is obtained, there is no need to reconstruct the entire data set. Accurate SIMPLISMA and ALS models were obtained from reduced data sizes of 4‰ using linear and 50 ppm using nonlinear compressions. For example, a set of six 35-mm slide images scanned at 1200 dpi producing nearly 34 Mbytes of data would compress to less than 175 kbytes. A single CD-ROM would hold the equivalent of nearly 24,000 slide images.[14]

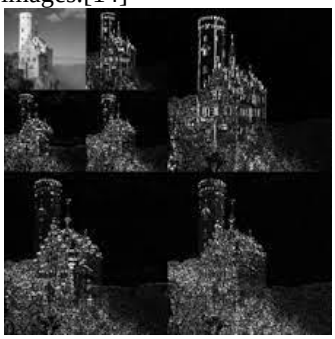


Illustration 12. Wavelets Compression Algorithm Process

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

Huffman coding provides codes to characters such that the length of the code depends on the relative frequency or weight of the corresponding character. Huffman codes are of variable-length, and without any prefix (that means no code is a prefix of any other). Any prefix-free binary code can be displayed or visualized as a binary tree with the encoded characters stored at the leaves.

Huffman tree or Huffman coding tree defines as a full binary tree in which each leaf of the tree corresponds to a letter in the given alphabet.

The Huffman tree is treated as the binary tree associated with minimum external path weight that means, the one associated with the minimum sum of weighted path lengths for the given set of leaves. So, the goal is to construct a tree with the minimum external path weight.

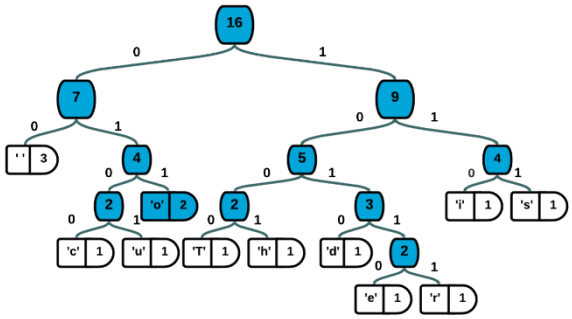


Illustration 13: Huffman tree generated from the exact frequencies of the text "This is our code".

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

Seam Carving

For the seam carving algorithm, we start by creating the energy map in which for each color channel, the energy is calculated by adding the absolute value of the gradient in the x direction to the absolute value of the gradient in the y direction. The energy for all color channels is summed into one 2D image to create the energy map. Then, an accumulated cost matrix must be constructed by starting at the top edge and iterating through the rows of the energy map. The value of a pixel in the accumulated cost matrix is equal to its corresponding pixel value in the energy map added to the minimum of its three top neighbors (top-left, top-center, and top-right) from the accumulated cost matrix. The minimum seam coordinates from Step 2 are then used to remove the minimum seam. All the pixels in each row after the pixel to be removed are shifted over one column. Finally, the width of the image has been reduced by exactly one pixel.

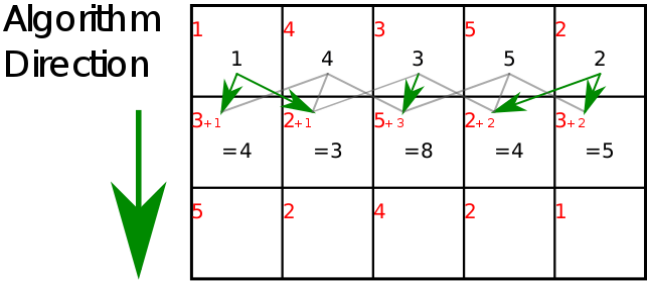


Illustration 14: Seam Carving using dynamic programming.

[1]https://github.com/sofiacaspul/ST0245-001/tree/master/proyecto

4.2.2 Lossless image-compression algorithm

Huffman Coding.

The Huffman code is a lossless compression algorithm that works by analyzing the frequencies that certain symbols appear in a message. For this we create a "dictionary" in which we store the frequencies of each data and create a priority queue with a minheap. Later, we implemented a binary tree where nodes store the dictionary, and sequentially merge the pairs of nodes that add up the lower frequencies, creating an intermediate node that stores that sum.
Symbols that appear most often will be encoded as a shorter bit string while symbols that are not used as much will be encoded as longer strings.
Finally, it is assigned the respective code for each symbol, each code is the sequence from the root to each character following the border labels, considering that the most frequently found symbols will be encoded as a shorter bit string, while the less used characters will have a longer encoding, seeking the maximum optimization

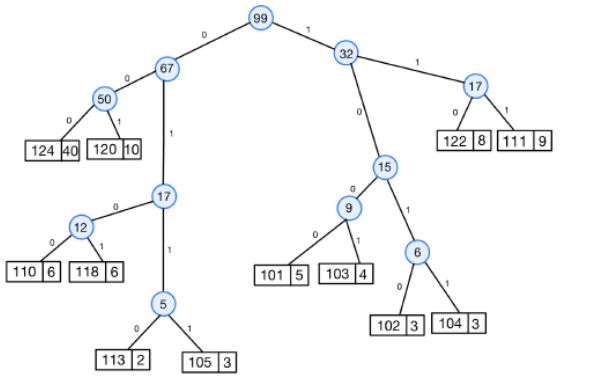


Illustration 15: Huffman tree simulation.

4.3 Complexity analysis of the algorithms

Algorithm	Time Complexity
Compression	O(M*N)
Decompression	O(M*N)

Table 1: Time Complexity of the Seam Carving image-compression and image-decompression algorithms. (M is the number of rows and N is the number of columns).

Algorithm	Memory Complexity
Compression	O(K)
Decompression	O(K)

Table 3: Memory Complexity of the image-compression and image-decompression algorithms. (Where k is the number of different pixels in the matrix).

5. RESULTS

5.1 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 4.

	Average execution time (s)	***Average file size (MB)***
Compression	19.6 s	0.36 MB
Decompression	0.34 s	0.36 MB

Table 4: Execution time of Seam Carving merged with Huffman coding and decoding both algorithms for different images in the data set.

5.2 Memory consumption

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

	Average memory consumption (MB)	***Average file size (MB)***
Compression	68.6 MB	0.36 MB
Decompression	70.6 MB	0.36 MB

Table 5: 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	3:1	3:1

Table 6: Rounded Average Compression Ratio of all the images of Healthy Cattle and Sick Cattle.

ACKNOWLEDGEMENTS

We appreciate the assistance of Mauricio Toro for the comments that helped to greatly improve this paper, in addition to all the constant recommendations given to fully carry out the project in the most efficient way and fulfilling the initial goal.

REFERENCES

1.Innovatione AgroFood Design. (2021, February 6). Ganadería de PRECISIÓN. Innovatione. https://innovatione.eu/2020/03/30/ganaderia-de-precision/.

2.Santillán, O. (n.d.). NOTAS INCYTU. 23. Ganadería de precisión. https://foroconsultivo.org.mx/INCyTU/index.php/notas/102-23-ganaderia-de-precision.

3.Kashiha, M. A., Bahr, C., Ott, S., Moons, C., Niewold, T., Tuyttens, F., & Berckmans, D. (2013). (rep.). Automatic monitoring of pig locomotion using image analysis.

4.Debauche, O., Mahmoudi, S., Andriamandroso, A. L. H., Manneback, P., Bindelle, J., & Lebeau, F. (2019). (rep.). Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors.

5.Shinde, T., & Prasad, J. (2017). (rep.). IoT based Animal Health Monitoring with Naive Bayes Classification.

6.Andrew, W., Burghardt, T., & Greatwood, C. (2017). (rep.). Visual Localization and Individual Identification of Holstein Friesian Cattle via Deep Learning.

7.YouTube. (2020). Seam Carving: Live Coding Session. YouTube. https://www.youtube.com/watch?v=ALcohd1q3dk&ab_channel=TheJuliaProgrammingLanguage.

8.Discrete cosine Transform (algorithm and Program). GeeksforGeeks. (2021, January 20). https://www.geeksforgeeks.org/discrete-cosine-transform-algorithm-program/.

9.Tabora, V. (2019, September 25). JPEG image scaling algorithms. Medium. https://medium.com/hd-pro/jpeg-image-scaling-algorithms-913987c9d588.

10.Fractal compression. Wikiwand. (n.d.). https://www.wikiwand.com/en/Fractal_compression.

11.Openspecs-Office. (2021, August 10). LZ77 compression algorithm. LZ77 Compression Algorithm | Microsoft Docs. https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-wusp/fb98aa28-5cd7-407f-8869-a6cef1ff1ccb.

12.Boyini, K. (2018, July 6). Huffman Coding Algorithm. Huffman coding Algorithm. https://www.tutorialspoint.com/Huffman-Coding-Algorithm.

13.LZW (LEMPEL–ZIV–WELCH) compression technique. GeeksforGeeks. (2019, September 9). https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/.

14.Ruckebusch, C. (2016). Resolving spectral mixtures: With applications from ultrafast time-resolved spectroscopy to super-resolution imaging. Elsevier.