Making Precision Livestock Farming More Energy Efficient Using Compression Algorithms

Alejandro Mc Ewen
Universidad Eafit
Colombia
amce@eafit.edu.co

Felipe Henao
Universidad Eafit
Colombia
fhenaog3@eafit.edu.co

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

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

Blue text = To complete for the 2nd deliverable

Violet text = To complete for the 3rd deliverable

ABSTRACT

Recently, the concept of precision livestock farming (PLF) has emerged, which has the focus of implementing information and communication technologies to improve the process of livestock farming. The problem is that once the data in farms can be digitalized, we need to find a way to make the system more energy efficient. Solving this problem is very important because it can improve the process of precision livestock farming, and thus improve the efficiency of livestock farming. Some related problems are the detection and individual identification of Holstein Friesian cattle, the use of an animal welfare platform for extensive livestock production, and the use of machine learning in precision livestock farming. 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

Farming is a very important in the economy and society. So, the more efficient the sector can be the better every is for it. If one can implement technology with farming a lot of process could be made more precise and efficient. This would help fulfill the need for food in the worlds growing population.

1.1. Problem

The problem is that farms don't have a great way to identify sick cattle. The only resource they can use is the famers intuition. With this unprecise method many cattle die because they aren´t treated. The problem is to device a more precise method to identify sick cattle.

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.

Explain, briefly, your solution to the problem *(In this semester, the solution is an implementation of compression algorithms. Which algorithms did you choose? Why?)*

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.

3.1 Computer and electronics in Agriculture

Farms needs to become more efficient to meet the food needs of a growing population. The problem is how can we make farming more efficient. The solution proposed by the article is to use precision livestock farming and machine learning to recognize healthy and not healthy cattle. The paper then does an in-depth investigation on various methods being used a research in this field. They did this with a Boolean search algorithm in various databases including Google Scholar, IIEE Xplore, Scopus, and Springer. After this they search for key words that satisfied the criteria that machine learning was being used for gracing, and that machine was being to enhance precision livestock farming in respects to the animal's health. This gave them 35 articles to use in their analysis. Finally, they ended by stating the limitation and challenges the industry has to solve to make this technology into a success.[10]

3.2 An Animal Welfare Platform for Extensive Livestock Production Systems

The article talks about the importance of keeping animal's welfare. Now more than ever people advocate for animals right and research has found that animal welfare correlates to people's welfare. The problem is how can one keep track of animal welfare. The researcher developed a collar that connects to a webserver to get the multiple variables like position and velocity for each animal. Then with this data they pass it through a machine learning, and it returns a well-being score. Finally, they state a few of the many application this idea has for example using them in cattle.[6]

3.3 Cloud Services Integration for Farm Animals' Behavior Studies Based on Smartphones as Activity Sensors

This article talks about the use of sensors in livestock farming, and the most critical parameter that can be monitored in animals is their behavior, since it provides information about their health and reproductive status. When analyzing an animal's behavior there are three main components, which are: the location obtained by radio frequency triangulation or GPS, the low frequency component of behavior as posture of the animal and the high frequency component of behavior. For this purpose, the use of smartphones was suggested as they are easily obtained, contain important sensors and don't require any hardware development. To monitor the animals' behavior, sensors like microphones, pressure sensors, electromyography, location sensors and accelerometers are used. In sensor data, all raw data is recorded with a decimal precision of 6 digits, so to compress this data they performed two things: remove redundancies and changing data to 3, 4 and 5 decimal digits. Once this data is reduced in volume, the data is stored and treated appropriately on the cloud. To store this data, the paper proposes an architecture structure which combines lambda architecture and a hosting and sharing platform. This platform separates data into video streams and is treated by streaming processing and time/event related data. Finally, they state that the lambda architecture proposed for collecting, storing, processing and sharing data between research teams can be used for other things different from cow behavior so that different teams can contribute to the system.[5]

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

This article talks about using computer vision pipelines with the aim to prove that Holstein Friesian cattle identification can be done automatically and non-intrusively. Traceability and identification of this cattle is highly demanded not only by exports and consumers demand but also countries introduced leally obligatory frameworks regarding national databases along with ear tagging, so the aim is to be able to improve in efficiency in the farm and contribute to animal welfare by making a non-intrusive and automated program to identify and trace these types of cattle. The first goal is to detect and locate Friesian cattle in images taken from a top-down or aerial standpoint, this is done by using the R-CNN (Regional Convolutional Regional Network) adaptation of the VGG CNN M 1024. Then, they used single frame individual identification and video based LRCN identification to individually identify the Friesian cattle. From this the results they obtained were near perfect results with a mean average precision of 99.3%, which proves that by standard deep learning pipelines, automatic Holstein Friesian detection and individual identification can be done. [1]

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 .

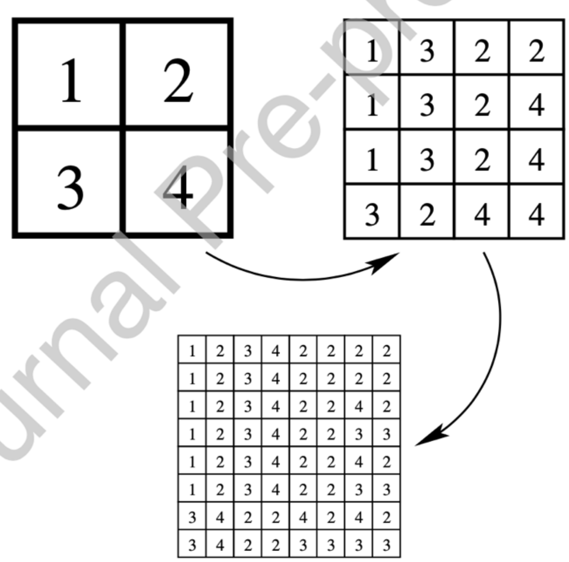
Finally, using the training data set, we trained a convolutional neural network for binary image-classification using Google Teachable Machine available at https://teachablemachine.withgoogle.com/train/image.

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 V-variable image compression

V-variable compression works by grouping pixel into a bigger pixel. It does this by making for groups of pixels using the k-means algorithm to separate them into as many groups as needed and then creating a bigger pixel combining the pixels into the most common pixel in the group. The complexity of the algorithm is linear because that is the time it takes to go and replace every pixel and run the k-means algorithm. This is how v-variable compression creates compressed images.[6]



Journal Pre-pro

3.2.2 Fractal Encoding

Fractal coding consists of removing repetitive structures in the image data, called fractals, from the image to be compressed. For this type of encoding, first we divide the input gray-scale image into blocks that do not overlap, these are called as Range blocks. For each R_i a Domain block (D_i) is selected so that this Domain block is similar to the Range block and twice the size. The complexity of fractal encoding is $O(n^4)$, since each range block is compared to all domain blocks.[8]

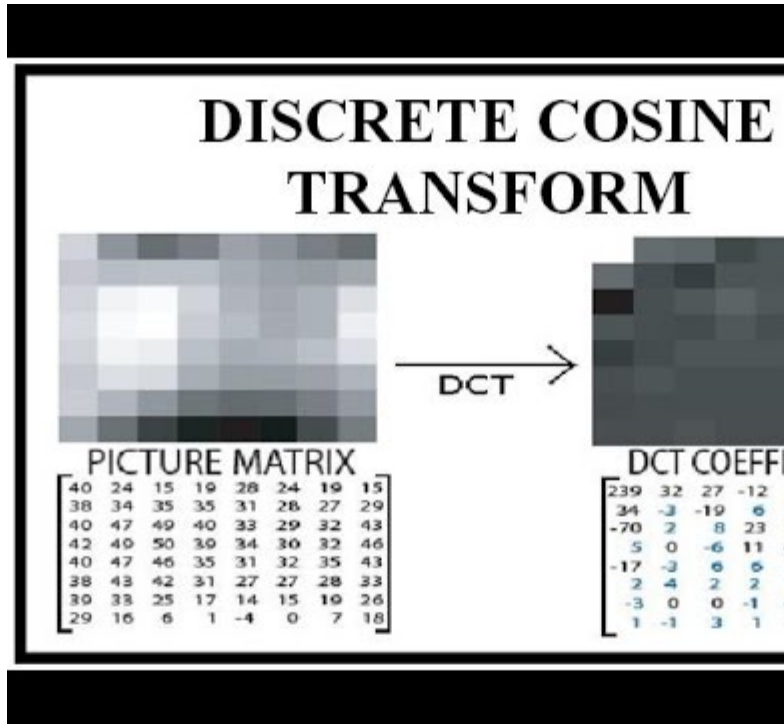
Figure 1 – In Partitioned Iterated Function System, self-sim
larger potions of the image (Domain blocks) and smaller potion

3.2.3 Discrete Cosine Transform (DCT)

Discrete Cosine Transform consists of representing the input image into a two-dimensional matrix, as P(i, j) and then, using the Discrete Cosine transform on the matrix, we obtain P'(i, j). Here, the image is divided into N blocks of size mxm, and is represented as a sum of cosine functions oscillating at different frequencies, following the formula:

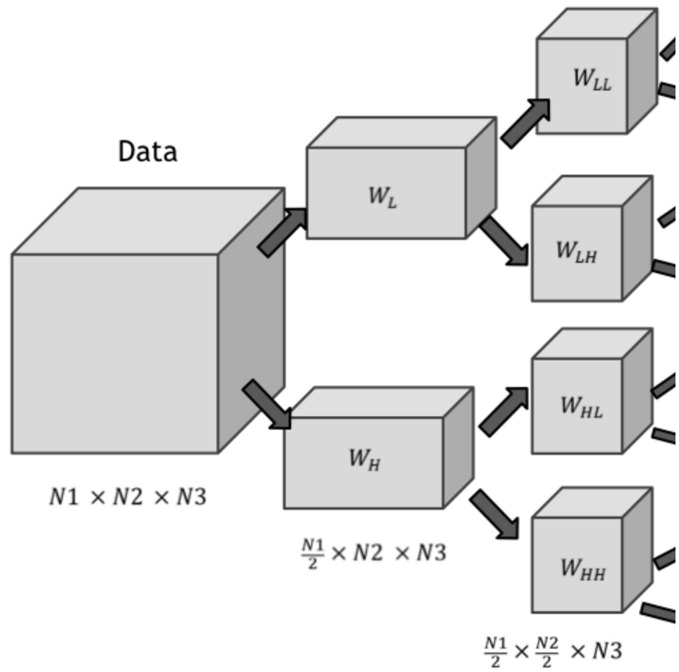
$$f(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \cos\left[\frac{\pi(2x+1)}{16}\right] \cos\left[\frac{\pi(2y+1)}{16}\right]$$

Were u=1,2..., N-1 and v=1,2..., N-1 and C(x) is $\frac{1}{\sqrt{2}}$ if x=0 or 1 if x $\neq 0$. Then, redundant image data is removed through quantization. The complexity of DCT is $O(n^2)$. [8]



3.2.4 Discrete Wavelet Transform

Discrete Wavelet Transform consists of dividing the input gray-scale image into four sub-bands represented as LL, LH, HL and HH, and placing watermark into one of these sub-bands. In this compression technique the image is processed in both dimensions by 2D-filters, which divide the image into four sub-bands LL1, LH1, HL1 and HH. The LL1 gives the harsher level of information, so it is divided again into LL1, LH2, HL2 and HH2, this division can be done up to 5 levels. Afterwards, quantization is used to obtain the wavelet coefficients, and these are used to get the compressed image through entropy encoding. The complexity of Discrete Wavelet Transform is O(n).[8]

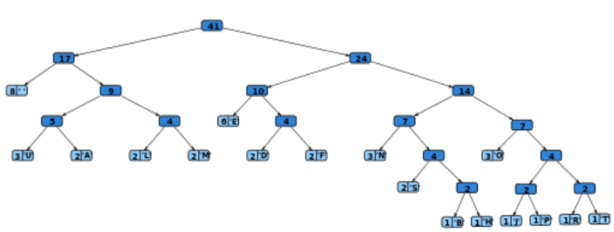


3.3 Lossless Image-compression alternatives

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 coding

The Huffman coding reassigns bit codes to different symbols depending on the rate they appear in an image. It does by putting each symbol on a binary tree ordering them the lower on the tree it is the less likely it is to appear. The complexity of this algorithm is linear because it only goes through the array twice first putting the values to the binary tree and then assigning the new bit codes to the characters on the binary tree. After, it goes through the image changing the bit codes.[3]



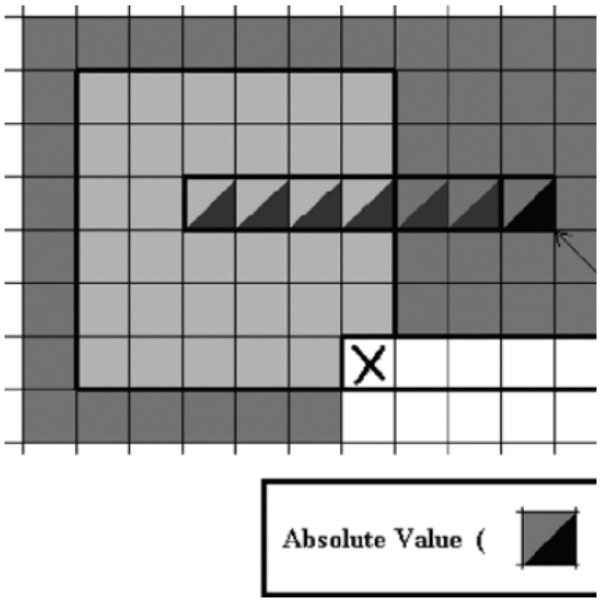
3.3.2 Burrows-Wheeler Transform

This Burrows-Wheeler Transform works by grouping similar characters together. This makes it easier to compress and require less space. It does this by rotating the string of characters and then selecting the option with the most repeated characters. Then it can use the output of the algorithm to get the original string. The complexity of this algorithm is linear because to make all the rotation it takes linear time or the length of the string. This is how the Burrows-Wheeler Transform algorithm works.[2]

Transformation				
1. Input	2. All rotations	3. Sort into lexical order	4. Take the last column	5. Output
^BANANA|	^BANANA| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	BNN^AA|A

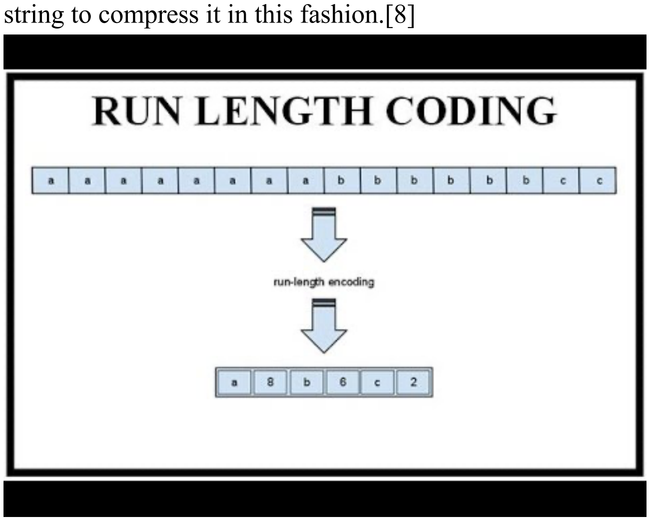
3.3.3 Grayscale two-dimensional Lempel-Ziv image compression scheme (GS-2D-LV)

In G2-2D-LV, an image is encoded by ordering the pixels by rows, which is known as raster scan order, processing one block of pixel each step. For each block of pixels, a similar match is located in the data that has already been encoded, if there is no suitable match the block of pixels is encoded using a prediction scheme. After the whole image is encoded, a statistical compression scheme is used to encode the match location, match position, residual and prediction error. The complexity of the GS-2D-LV scheme is linear.[9]



3.3.4 Run Length Encoding (RLE)

Run length encoding is compressing an image by grouping equal values into the value and a counter. This allows for images with a lot of repeat values in a row that are the majority to be made into a much shorter string. This coupled with other methods like Burrows-Wheeler Transform can make the image even shorter. The complexity of this algorithm is linear because it only takes one look through the string to compress it in this fashion.[8]



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

Explain the data structure used to make the image compression and make a figure explaining it. Do not use figures from the Internet. *(In this semester, example of the data structures are trees and hash tables)*

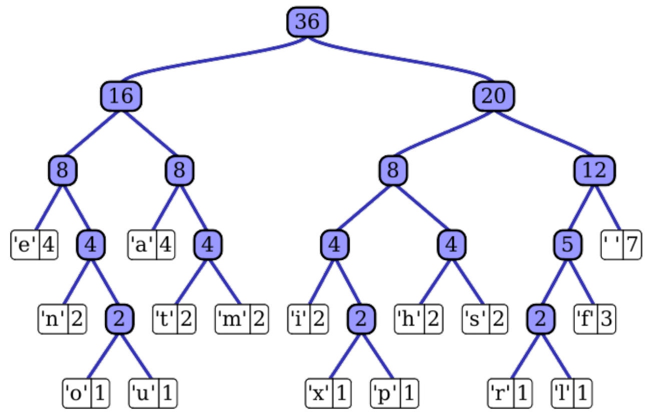


Figure 1: Huffman tree generated from the exact frequencies of the text "this" (Please, feel free to change this Figure if you use a different data structure).

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.

Explain the design of the algorithms to solve the problem and make a figure. Do not use figures from the Internet, make your own. *(In this semester, one algorithm must be a lossy image-compression algorithm such as image scaling, seam carving or wavelet compression and the second algorithm must be a lossless image-compression algorithm such as Huffman coding, LZS or LZ77).*

4.2.1 Lossy image-compression algorithm

Explain, briefly, how did apply a lossy image-compression algorithm such as seam carving or image scaling. Explain also decompression.

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

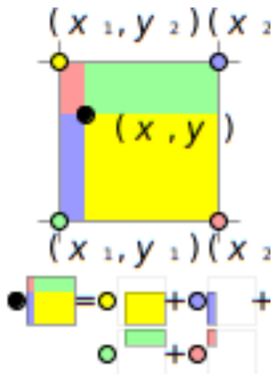


Figure 2: Image scaling using bi-lineal interpolation. (Please, feel free to change this Figure if you use a different data structure).

4.2.2 Lossless image-compression algorithm

Explain, briefly, how did you apply a lossless image-compression algorithm such as Huffman coding, LZS or LZ77. Explain also decompression.

4.3 Complexity analysis of the algorithms

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.

To measure memory consumption, you should use a profiler. A very good one for Java is VisualVM, developed by Oracle, http://docs.oracle.com/javase/7/docs/technotes/guides/visualvm/profiler.html. For Python, use C Profiler.

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 over-fitting? 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] Andrew, W., Greatwood, C., & Burghardt, T. 2017. Visual localisation and individual identification of holstein friesian cattle via deep learning. Retrieved February 14, 2021 from https://openaccess.thecvf.com/content_ICCV_2017_workshops/w41/html/Andrew_Visual_Localisation_and_ICCV_2017_paper.html

[2] Anon. 2021. Burrows–Wheeler transform. (February 2021). Retrieved February 14, 2021 from https://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform

[3] Anon. 2021. Codificación Huffman. (January 2021). Retrieved February 14, 2021 from https://es.wikipedia.org/wiki/Codificaci%C3%B3n_Huffman

[4] Anon. 2020. Run-length encoding. (December 2020). Retrieved February 14, 2021 from https://en.wikipedia.org/wiki/Run-length_encoding

[5] Debauche, O., Mahmoudi, S., Andriamandroso, A.L.H. et al. 2019. Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors. J. Retrieved February 14, 2021 from

https://doi.org/10.1007/s12652-018-0845-9

[6] Doulgerakis, Vasileios & Kalyvas, Dimitrios & Bocaj, Enkeleda & Giannousis, Christos & Feidakis, Michalis & Laliotis, George & Patrikakis, Charalampos & Bizelis, Iosif. 2019. An Animal Welfare Platform for Extensive Livestock Production Systems. Retrieved February 14, 2021 from https://www.researchgate.net/publication/338595895_An_Animal_Welfare_Platform_for_Extensive_Livestock_Production_Systems

[7] Franklin Mendivil, Örjan Stenflo. 2020. Extreme compression of grayscale images. Communications in Nonlinear Science and Numerical Simulation, Vol 9. Retrieved February 14, 2021 from https://doi.org/10.1016/j.cnsns.2020.105546.

[8] Kulkarni, A., & Junnarkar, A. (2015). Gray-Scale Image Compression Techniques: A Review. Retrieved February 14, 2021 from

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.1396&rep=rep1&type=pdf

[9] Nathanael J. Brittain, Mahmoud R. El-Sakka. 2007. Grayscale true two-dimensional dictionary-based image compression. Journal of Visual Communication and Image Representation, Vol 18. Retrieved February 14, 2021 from https://doi.org/10.1016/j.jvcir.2006.09.001.

[10] Rodrigo García, Jose Aguilar, Mauricio Toro, Angel Pinto, Paul Rodríguez. 2020. A systematic literature review on the use of machine learning in precision livestock farming. Computers and Electronics in Agriculture, Vol 179. Retrieved February 14, 2021 from

https://doi.org/10.1016/j.compag.2020.105826.

[11] Anon. 2020. Discrete Cosine Transform. (December 2020). Retrieved February 14, 2021 from https://en.wikipedia.org/wiki/Discrete_cosine_transform