**USE OF NEURAL NETWORKS AND COMPRESSION ALGORITHMS FOR COW’S -HEALTH CLASSIFICATION**

|  |  |  |  |
| --- | --- | --- | --- |
| Victor Daniel Agudelo Vallejo  Universidad Eafit  Colombia  vdagudelov@eafit.edu.co | Daniel Ricardo Palacios Diego  Universidad Eafit  Colombia  drpalaciod@eafit.edu.co | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

# **ABSTRACT**

In this report, we show how we will approach the cow’s health classification problem using machine learning and image compression. Usually, in livestock farms internet infrastructure is poor, therefore, Before passing the images through AI we must compress them as much as possible without losing quality and wasting resources.

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**

*Our motivation is to take agriculture to another level. We want to facilitate the breeding and fattening process. To do that we will develop a algorithm that processes the pictures of the animals and determines if it is sick or good condition. During the past century, agricultural production was focused mainly on human needs coverage, price and competition. However, in recent decades, consumers expect their food to be produced and processed with greater. respect towards animal welfare [MW14], as consumer health has been closely linked to the welfare of animals. The farms lately have been taking more care of livestock health, to avoid lawsuits. In a small farm it's easy to control a sick cow But in a big farm it can be difficult to keep track of all the animals and that's why this software is needed to analyze each animal one by one and determine its status.*

# **1.1. Problem**

In modern livestock farming, one of the most important matters is to detect diseases and pests on livestock and treat them as fast as possible. Farmers often do this process manually for thousands of animals, which is inefficient and expensive. We have to make this process more efficient, fast, and resource-friendly, so the farmers can focus more on treating the disease rather than trying to detect it.

**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. but we compress the images to the best quality possible, so that the farmer can enlarge or see the image from any point of view and see exactly the problem. The proposed compression algorithms are a lossy compression one, seam carving and a lossless compression one, Huffman coding That's why our algorithm is the solution to this problem. The algorithm compresses the images using few resources.

**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.Explain four (4) articles related to the problem described in Section 1.1. You may find the related problems in scientific journals. Consider Google Scholar for your search. *(In this semester, related work is research on animal-health classification and data compression, in the context of PLF).*

**3.1 ANIMAL CLASSIFICATION USING TRANSFER LEARNING**

## Training a neural network from scratch for animal classification needs a large amount of data sets and intensive training which require a lot of computing power and time, so for solving this problem Man hu et al[1] used the transfer learning method so they could use a pre-trained neural network (Resnet18) and adapt it for animal classification. They were able to significantly increase recognition accuracy and speed compared to the non-transfer learning model.

## **3.2 LIVESTOCK DETECTION IN AN AERIAL IMAGE USING MACHINE LEARNING**

They improved the RCNN and Yolo-v3 algorithm, giving more resolution to the aerial pictures and containing livestock with varying shapes, scales, and orientations. Having an appropriate number of livestock on grassland is important for sustainable animal husbandry. If the stocking rate is too high, the grassland will be overutilized, and the grassland ecology will deteriorate, which is not conducive to the production of livestock. Accurately determining the actual number of livestock present on grassland is necessary

for macro control, and is needed for government surveillance of overgrazing to prevent grassland ecological degradation therefore, an efficient and accurate method for detecting livestock is needed to obtain the actual number of animals grazing in the grassland. The convolutional neural networks have been used widely for computer vision tasks, including image classification, object recognition, and semantic segmentation. but object detectors based on conventional datasets do not perform well on aerial images, the main reason being that aerial images have their own particularity. This algorithm edits the RCNN and Yolo-v3 to have more precision in the pictures and count all the animals that are in the grassland including the cow´s sons. with more precision. Each target has a resolution of between 20 × 20 and 40 × 40 pixels. They have varying colors and random.They use U-net to score each pixel in the image, and preserve regions with high scores as regions of interest. Instead of using a fixed threshold score for selecting ROIs, they map the score of each pixel to 0–255, as a grayscale image, and then use an adaptive threshold segmentation method to segment the ROIs.

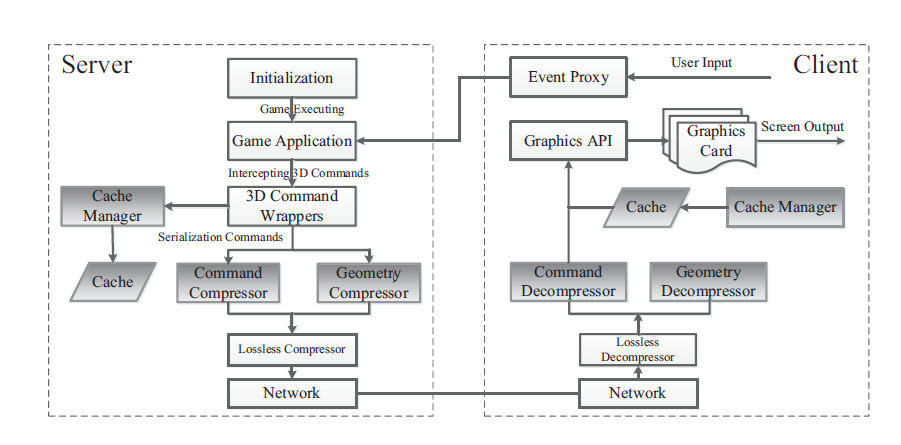
## **3.3 Image Compression in VideoGames**

This problem is about video games is based in why do old games look so bad on moderns tvs, there are few problems:

* bloom phosphor effect
* Scanlines
* 240 signals sometimes aren't even supported on modern tvs.

they found some solutions to this problem:

* Digital downloads
* Handle upscaling on the game console
* Emulation

In cloud gaming systems, the game program runs at servers in the cloud, while clients access game services by sending input events to the servers and receiving game scenes via video streaming. Game execution occurs at the servers, and graphics commands are captured and transmitted to clients for screen renderings. Its main limitation is that it is very bandwidth consuming. 

## **3.4 CLOUD SERVICES INTEGRATION FOR FARM ANIMALS’ BEHAVIOR STUDIES BASED ON SMARTPHONES AS ACTIVITY SENSORS**

The iPhone is an inexpensive means of measuring cow behavior. I-Phone

5SE, 6S, 7S and 8S are equipped with a new factory calibrated IMU. This new IMU is not much evaluated in

scientific literature (Yang 2017a, b). Currently the data is

transmitted from the iPhone to the gateway by using the

UDP protocol on WIFI, but this protocol may cause a data

packet loss problem when it is necessary to collect data from several iPhone simultaneously. The compressibility of data massively acquired can be reduced by 43.5% on average and this can hardly be improved any further. By opposition, they have shown that individual parameters can be highly compressible. In the future when the most explicative parameters will be selected for a given research application, the compressibility of data will be improved. Finally, applying edge computing during the massive collection of data is not interesting. The future development of microcontrollers which acquire pertinent parameters at specific sampling rate and use low throughput LoRa will resolve this issue. Indeed, using acknowledgment in the protocol LoRA guarantees the correct reception of payload transmitted. The Lambda architecture proposed for collecting, storing, processing and sharing data between research teams is flexible enough to be used for other uses than cow behavior provided that different teams contribute to the system. Other data compression algorithms must be considered to optimize the energy consumption of the battery. The sharing platform will integrate a billing system to evaluate on one hand the using of data and multi-tenancy of software on the other hand. Each segment is composed of 5–10 million stamped events that covers one period of time. Segments can be compressed by LZ45 by default or LZF6 algorithm and can also be stored in a column orientation database. Druid cluster is composed of 4 kinds of nodes: real-time, historical, broker and coordinator.

## **3. MATERIALS AND METHODS**

In this section, we explain how the data was collected and processed and, after, different image-compression algorithm alternatives to 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/dataset](https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets)s .

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).*

**3.2.2 Seam carving**

A seam is a path of pixels that share edges or sides that cross the image, they can be vertical or horizontal.

Seam carving is really interesting because you can also use it for enlarging images, the idea is to find seams with low energy pixels that are less important to the image, so they can be eliminated or re added depending if we want to shrink or enlarge the size of the image.

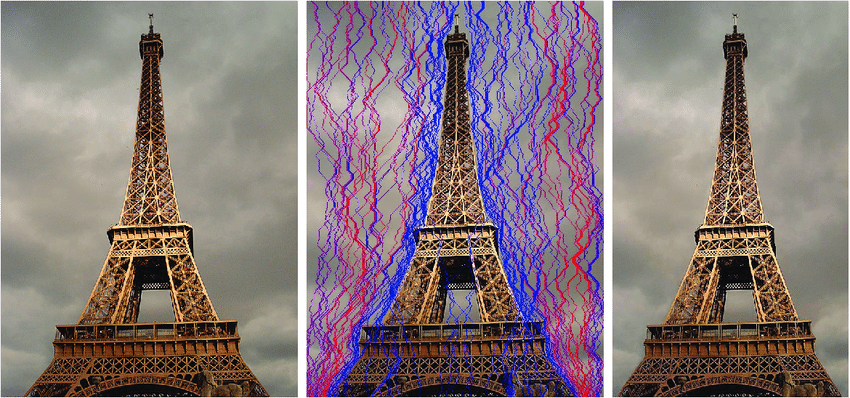


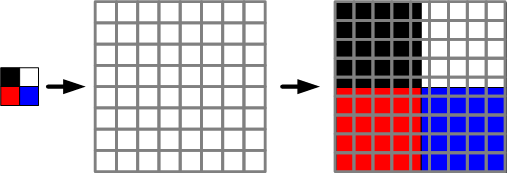
Image from [6]

Using dynamic programming this algorithm time complexity is found to be O(N\*M) M and N being the width and the height of the original image respectively[5].

**3.2.3 Nearest neighbour interpolation**

This image scaling algorithm is really easy to implement but the more we upscale the image the more “pixelated” it will look like, so it’s not recommended if the image quality is important.

This algorithm simply replaces all the pixels in the output with the nearest pixel in the input, which means for upscaling multiply the number of pixels from the original photo. It’s complexity is really low compared to other interpolation algorithms

**4. ALGORITHM DESIGN AND IMPLEMENTATION**

**3.2.4 Discrete cosine transform**

The discrete cosine transform (DCT) represents an image as a sum of sinusoids of varying magnitudes and frequencies.

## Most of the visually significant information about an image is concentrated in just a few coefficients of the sum of sinusoids, so it is used in one of the most famous compression methods, the jpeg.

## **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).*

**3.2.1 L277 and lZ78**

LZ77 and LZ78 are the two lossless data compression algorithms published in papers. These two algorithms form the basis for many variations including LZW, LZSS, LZMA and others. Besides their academic influence, these algorithms formed the basis of several ubiquitous compression schemes, including GIF and the deflate algorithm used in PNG and ZIP.

## **3.3.2 Burrows Wheeler Transform**

This algorithm is often used for text compression but it can be implemented for image compression too.

## The idea of this algorithm is given a string to get all the rotations of the string, a rotation is another string from taking one end of the original string and putting it at the other end. Then we must sort lexicographically the list of rotations and get the characters at the last index.

## **3.3.3 Lempel-Ziv-Stac(LZS)**

## LZS compression and decompression uses an LZ77 type algorithm. It uses the last 2 KB of uncompressed data as a sliding-window dictionary.

## An LZS compressor looks for matches between the data to be compressed and the last 2 KB of data. If it finds a match, it encodes an offset/length reference to the dictionary. If no match is found, the next data byte is encoded as a "literal" byte. The compressed data stream ends with an end-marker.[7]

## 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]](#footnote-1).

## **4.1 Data Structures**

Implementation Huffman Code: To implement Huffman Encoding, we start with a Node class, which refers to the nodes of Binary Huffman Tree. In that essence, each node has a symbol and related probability variable, a left and right child and code variable. Code variable will be 0 or 1 when we travel through the Huffman Tree according to the side we pick (left 0, right 1). We have 3 helper functions, first one for calculating the probabilities of the symbols in the given data, second one for obtaining encodings of symbols which will be used after having Huffman Tree and the last one for obtaining output. Additionally, we have a Total Gain function which takes the initial data, and the dictionary comes from Calculate Code holding the symbols and their codes together. That function calculates the difference between the bit size of compressed and non-compressed data. Finally, we have Huffman Encoding function which takes only the data as parameter and gives us the result encoding and the total gain using all these 4 functions.

To decode the encoded data, we require the Huffman tree. We iterate through the binary encoded data. To find character corresponding to current bits, we use following simple steps.

1. We start from root and do following until a leaf is found.
2. If current bit is 0, we move to left node of the tree.
3. If the bit is 1, we move to right node of the tree.
4. If during traversal, we encounter a leaf node, we print character of that leaf node and then again continue the iteration of the encoded data starting from step 1.

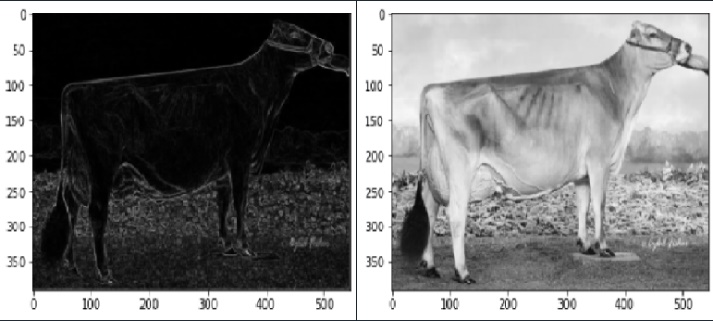


**Figure 1:** Huffman tree generated from the exact frequencies of the text "this” (Please, feel free to change this Figure if you use a different data structure).

**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.

**4.2.1 Lossy image-compression algorithm**

****

**Figure 2:** The left image is an energy map for the right image data structure).

We decided to use seam carving as the lossy compression algorithm for this project. We can explain how it works in few steps

1. We calculated the energy of every single pixel using the dual gradient function, where we calculate the x and y gradient using the horizontal and vertical neighbor of the pixel respectively, the result will be a energy map such as the one shown in figure 2

The most important part of the algorithm is finding the lowest energy seams, a seam is a line of pixels that goes from top to bottom or left right throughout the image.

1. Using dynamic programming we find the seam with the lowest energy from top to the bottom of the image
2. Finally, we erase that seam from the original image and update it

For our main purpose, we compress 25% of the image, vertically and horizontally, for horizontal seam removal we first must rotate the image 90° and procced as normal, due to our algorithm essentially can only find vertical seams, so we have to do this small trick.

**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(N2\*M2) |
| Decompression | O(N3\*M\*2N) |

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

|  |  |
| --- | --- |
| **Algorithm** | **Memory Complexity** |
| Compression | O(N\*M\*2N ) |
| Decompression | O(2M\*2N) |

**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.Man Hu and Fucheng You. 2020. Research on animal image classification based on transfer learning. In Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering (EITCE 2020). Association for Computing Machinery, New York, NY, USA, 756–761.

2. Cheng, M.-M.; Zhang, F.-L.; Mitra, N. J.; Huang, X.; Hu, S.-M. RepFinder: Finding approximately repeated scene elements for image editing. ACM Transactions on Graphics Vol. 29, No. 4, Article No. 83, 2010.

3. X. Liao et al., "LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming," in IEEE/ACM Transactions on Networking, vol. 24, no. 4, pp. 2128-2139, Aug. 2016, doi: 10.1109/TNET.2015.2450254.

4 .Yang F, Tschetter E, Léauté X, Ray N, Merlino G, Ganguli D (2014) A real-time analytical data store. SIGMOD’14, June 22–27, 2014, Snowbird, UT, USA. ACM 978-1-4503-2376-5$414/06. https:// doi.org/10.1145/2588555.2595631

5.Aditya Sharma.Image Compression using Seam Carving and Clustering,*2019, retrieved in july 16th 2021:*[*https://adityashrm21.github.io/Image-Compression/*](https://adityashrm21.github.io/Image-Compression/)

*6. Kopf, Stephan & Guthier, Benjamin & Lemelson, Hendrik & Effelsberg, Wolfgang. (2009). Adaptation of Web Pages and Images for Mobile Applications. retrieved in july 16th*

*7. Wikipedia. 2017. Lempel-Ziv-Stac.2019. retrieved in july 16th*

1. https://github.com/DanielPalacios05/ST0245-001 [↑](#footnote-ref-1)