Algoritmos de compresión de imágenes para evaluar la salud animal del ganado en el contexto de la Ganadería de Precisión

Jerónimo Osorio Muñoz
Universidad Eafit
Colombia
josoriom2@eafit.edu.co

Jorge Alfredo Villarreal Márquez
Universidad Eafit
Colombia
javillarrm@eafit.edu.co

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

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

RESUMEN
Durante los últimos tiempos el mantenimiento de la ganadería se ha vuelto un proceso más masivo y por lo tanto más complejo, por lo que la automatización de sus procesos de cuidado suponen una gran ventaja a la hora de reducir el tiempo de procesos que suponen gastos logísticos si se hacen de una manera no computacional. Este trabajo busca esbozar lo mencionado anteriormente mediante la compresión de imágenes de ganado tanto enfermo como sano con el fin de clasificar la salud de un animal N en el contexto de la ganadería de precisión. Con la importancia del problema y su definición es propicio mencionar algunos problemas que atañen a este contexto como lo es la identificación temprana de posibles individuos enfermos que puedan contagiar a los demás, o el constante tanteo de individuos sanos para fines de identificar el potencial de ganancias en un periodo de tiempo determinado, como también la separación por lotes de vacunos yendo de los que están en peores condiciones a los que están en mejores, entre otros problemas. Se obtuvieron alrededor de 0.1 s/MB y se consumen alrededor de 240 MB en compresión 2:1 de imágenes.

Palabras clave
Algoritmos de compresión, aprendizaje de máquina, aprendizaje profundo, ganadería de precisión, salud animal.

1. INTRODUCCIÓN

La Ganadería de Precisión (GdP) es un conjunto de tecnologías que buscan el monitoreo y manejo en tiempo real del ganado. Esto es necesario porque el sector agropecuario experimenta aumento en la demanda mientras hay un declive en la cantidad de trabajadores del sector. Para garantizar la salud y producción del ganado, hay que mitigar los efectos de la escasez de trabajadores. Para ello, es vital desarrollar tecnologías que equilibran altas tasas de transmisión y procesamiento de datos con electrónicos eficientes y de bajo costo para uso en áreas grandes con poca cobertura. [1] Por ello nuestro proyecto busca mediante la compresión de imágenes agilizar estos procesos y por lo tanto disminuir costos tanto de tiempo como monetarios.

1.1. Problema

Mediante la compresión de imágenes se busca automatizar el proceso de clasificación del estado de salud del ganado con el fin

de ayudar a la optimización de tiempos y de costos en los procesos de la ganadería de precisión.

2. TRABAJOS RELACIONADOS

En lo que sigue, explicamos cuatro trabajos relacionados. en el dominio de la clasificación de la salud animal y la compresión de datos. en el contexto del PLF.

2.1 Compresión eficiente para Wireless Sensor Network

Se investigó la eficiencia de compresión sin pérdidas de una versión ajustada de LZW, para optimizar la eficiencia de batería en una red de sensores sin cables. Además, se comparó con bzip2 y gzip. El algoritmo obtuvo una razón de compresión de 76.47% para datos de electrocardiograma, en un algoritmo más simple que la competencia [2].

2.2 Compresión en tiempo real de baja complejidad para sistemas multicámara

Usando correlaciones espaciales de sistemas distribuidos de cámaras, se diseñó e implementó una solución en tiempo real a través de Distributed Dictionary Learning, un tipo de compresión por machine learning de la familia del aprendizaje de características. Las métricas para comparación fueron el error cuadrático medio y la “precisión” [3].

2.3 Compresión de baja complejidad y energía para WSN

Proponen un algoritmo de pérdida basado en LBT con PSNR comparable en diferentes bitrates y siendo apto para redes de sensores sin cableado. También encuentran mejoras en términos del tiempo útil de los dispositivos por la baja complejidad [4].

2.4 Monitoreo de temperatura de ganado sin contacto

Con modelos de machine learning, se evaluó un sistema para evaluar cambios en la temperatura interna del ganado. Esto permite la detección de condiciones anormales antes de la aparición de síntomas severos en animales de manera simple. También se propone un sistema de cámaras térmicas para evitar el contacto.[6]

3. MATERIALES Y MÉTODOS

En esta sección, explicamos cómo se recogieron y procesaron los datos y, después, diferentes alternativas de algoritmos de compresión de imágenes para mejorar la clasificación de la salud animal.

3.1 Recopilación y procesamiento de datos

Recogimos datos de *Google Images* y *Bing Images* divididos en dos grupos: ganado sano y ganado enfermo. Para el ganado sano, la cadena de búsqueda era "cow". Para el ganado enfermo, la cadena de búsqueda era "cow + sick".

En el siguiente paso, ambos grupos de imágenes fueron transformadas a escala de grises usando Python OpenCV y fueron transformadas en archivos de valores separados por comas (en inglés, CSV). Los conjuntos de datos estaban equilibrados.

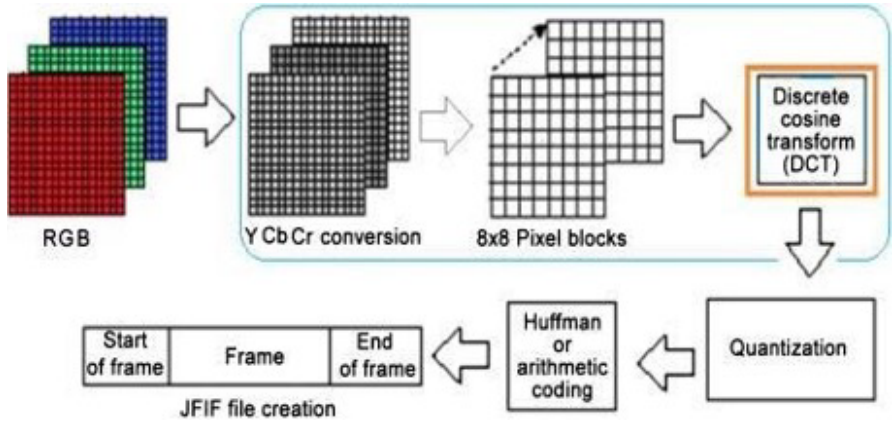
El conjunto de datos se dividió en un 70% para entrenamiento y un 30% para pruebas. Los conjuntos de datos están disponibles en
https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets .

Por último, utilizando el conjunto de datos de entrenamiento, entrenamos una red neuronal convolucional para la clasificación

binaria de imágenes utilizando *Teachable Machine* de Google disponible en https://teachablemachine.withgoogle.com/train/image.

3.2 Alternativas de compresión de imágenes con pérdida

A continuación veremos diferentes alternativas de compresión de imágenes con pérdida:



[8]

3.2.1 Transformación de coseno discreto

Discrete cosine transform

Este método es muy importante en el proceso de compresión de imagen/video porque cuenta con una energía de compactación muy alta. Es un método que hace muchos cálculos aritméticos para que la transformación en tiempo real sea eficiente

El algoritmo va de la siguiente manera, primero pasa de rgb a conversión Y CB Cr después pasa a una matriz de píxeles 8x8 y después se hace la transformación por transformación de coseno discreto.[7]

3.2.2 Compresión fractal

La compresión fractal es un método para la compresión de imágenes basada en fractales, este método usa un algoritmo para convertir las partes en “fractal codes”los cuales se usan para recrear la imagen comprimida. **[9]**

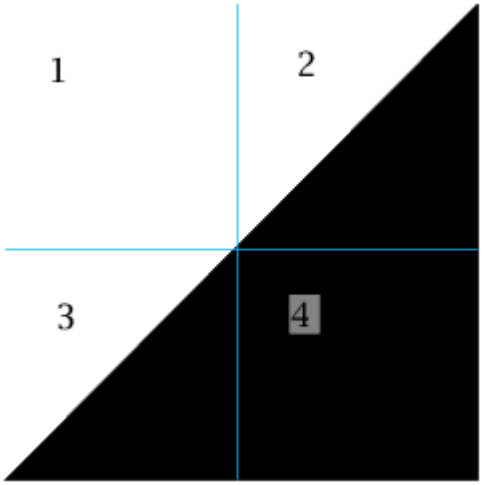
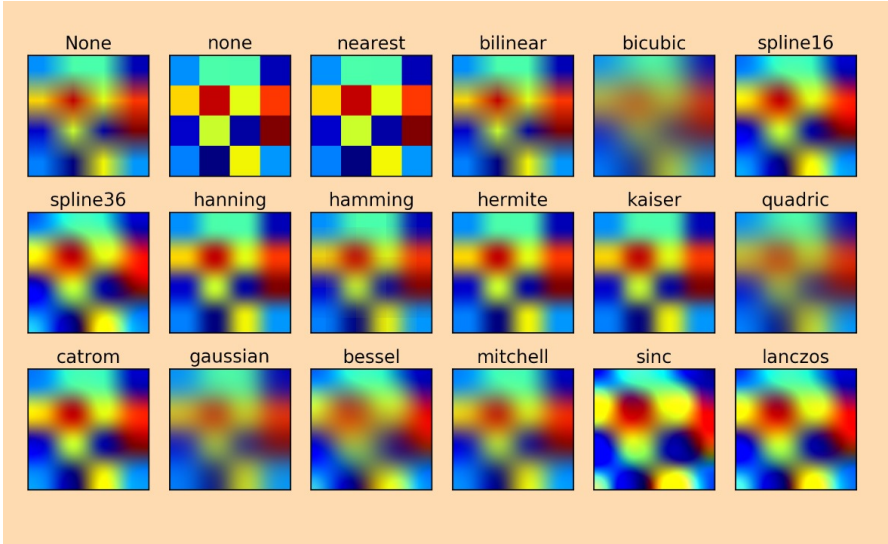


Imagen del método de compresión fractal [10]

3.2.3 Escalado de imágenes

Este método se refiere al reordenamiento del tamaño de una imagen, dependiendo de la imagen puede ser escaladas usando diferentes tipos de transformaciones geométricas o crear una nueva con más o menos número de píxeles

En este método existen 3 algoritmos comunes de escalamiento: 1. Nearest Neighbor Scaling. 2.Bilinear y 3. Bicubic Interpolation **[11]**



[12]

3.2.4 Enhanced Compressed Wavelet

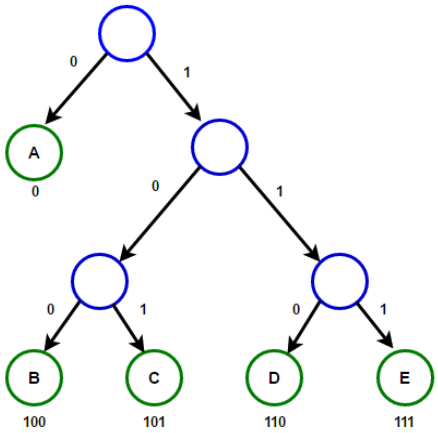
Este algoritmo tiene unos ratios de compresión muy elevados, que van desde 10:1 hasta de 50:1, mediante el uso de técnicas de ondículas. Gracias a esto se reduce considerablemente el tamaño de los archivos, lo que permite una rápida compresión y descompresión mediante un uso escaso de la memoria RAM. Este método comprime transformando las imágenes al espacio wavelet usando la Transformada Discreta de Wavelet. Se reduce la cantidad de información de la imagen mediante cuantización,esto para pasarlas a Wavelet. La imagen comprimida ECW se procesa línea a línea directamente a partir de la imagen original. La técnica de compresión ECW puede comprimir imágenes de cualquier tamaño usando un algoritmo

recursivo de segmentación que no precisa del uso de almacenamiento en disco mientras se realiza.[13]

3.3 Alternativas de compresión de imágenes sin pérdida

3.3.1 Codificación de Huffman

Para reducir espacio, se asignan códigos de longitud óptima a los datos de entrada según la cantidad de ocurrencias de cada dato de entrada. Para lograr esto, primero se construye un árbol de Huffman a partir de los datos de entrada, los cuales se etiquetan de nodos de hoja como símbolos abstractos. Luego empieza a sumar sucesivamente los nodos con menor probabilidad hasta alcanzar un único nodo. Por último, se viaja en el árbol asignando códigos a cada nodo, según su probabilidad, teniendo los más frecuentes códigos más cortos.

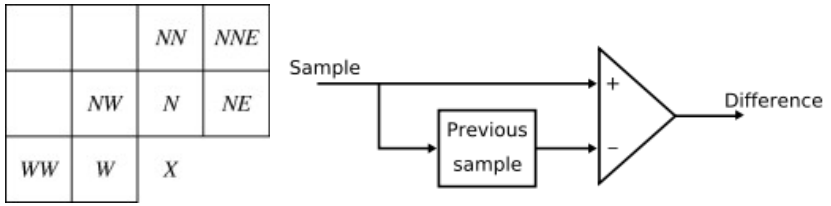


[17]

Por favor, explique el algoritmo, su complejidad e incluya una figura vectorial.

3.3.2 CALIC

En Context Adaptive Lossless Image Compression (CALIC), dado que los pixeles que son cercanos generalmente también tienen valores cercanos, cuando se usa en imágenes en escala de grises, usa la información de que tanto cambian los valores de píxeles en una muestra de un área respecto a un pixel para generar predicciones en forma de vectores que se complementan entre sí con cada área adyacente. Esto también permite la detección primitiva de bordes en una imagen. También se tiene que restar las predicciones con los datos verdaderos para mantener el error de predicción en cuenta para refinar la predicción. Este método resalta por usar tanto contexto como predicción en el área de medicina para comprimir imágenes en escala de grises [14].

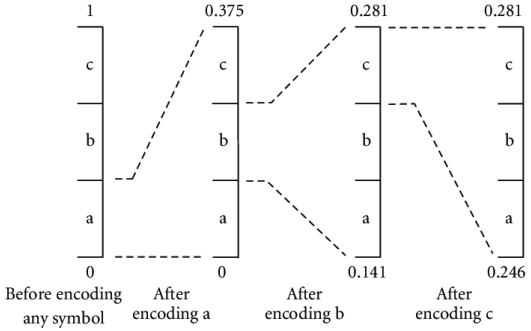


[14][15]

3.3.3 Codificación Aritmética

Es similar a la codificación de Huffman, pero en vez de construir un árbol de Huffman para símbolos abstractos a los que se

determina su frecuencia para asignarles códigos, se piensa en las frecuencias de números naturales que se asignan a los datos. Estos números, a su vez, se pueden dividir hasta llegar hasta un único número de precisión arbitraria entre 0 y 1 en vez de la raíz del árbol.[16]



3.3.4 Run Length Encoding (RLE)

Este método es óptimo cuando en una imagen es común encontrar información redundante y consecutiva, por ejemplo en imágenes en escala de grises. Primero se guardan los valores de la matriz de la imagen en un vector, luego, cada dato repetitivo de imagen, es reemplazado por una secuencia de {d,C}, siendo los d datos repetidos y C el número de ocurrencias. Al descomprimir simplemente se reemplaza cada d un número C de veces. Si bien funciona bien en imágenes en blanco y negro por su rango de valores limitados hay que tener en cuenta que no es óptimo cuando no hay muchos datos repetidos, como cuando hay ruido [18] .

4. DISEÑO E IMPLEMENTACIÓN DE LOS ALGORITMOS

En lo que sigue, explicamos las estructuras de datos y los algoritmos utilizados en este trabajo. Las implementaciones de las estructuras de datos y los algoritmos están disponibles en Github[1].

4.1 Estructuras de datos

Explique la estructura de datos utilizada para hacer la compresión de las imágenes y haga una figura que la explique. No utilice figuras de Internet. *(En este semestre, ejemplo de las estructuras de datos son los árboles y las tablas hash)*

4.1.1 Numpy ndarray

El numpy ndarray es un contenedor n-dimensional de elementos de cantidad fija con un mismo tipo y longitud. Sin embargo, bajo el uso que se le da en este trabajo, para representar la matriz de cada imágen. Por eso, su complejidad será igual a la de un arreglo de arreglos de igual longitud, que se guardan como bloques contiguos de memoria.

4.2 Algoritmos

En este trabajo, proponemos un algoritmo de compresión que es una combinación de un algoritmo de compresión de imágenes con pérdidas y un algoritmo de compresión de imágenes sin pérdidas. También explicamos cómo funciona la descompresión para el algoritmo propuesto.

.

4.2.1 Algoritmo de compresión de imágenes con pérdida

El algoritmo que escogimos es una implementación propia del nearest neighbor que consiste en comprimir la imagen mediante una constante de cambio que define la escala en la que se quiere reducir las dimensiones de la imagen, esto funciona mediante las posiciones de los valores numéricos del csv en la matriz de píxeles, como lo demostramos matemáticamente a continuación:

(Las matrices se hicieron a mano en word)

Tenemos una matriz original 8x8 que responde a los datos csv de la imagen

220	192	204	207	198	205	203	202
185	140	137	139	134	139	133	132
203	140	124	128	128	132	125	125
211	143	127	137	139	141	136	142
194	134	127	138	136	135	132	142
197	149	149	156	148	146	142	149
203	160	161	163	156	161	155	155
206	165	164	165	163	175	170	163

Procedemos a expresar sus posiciones

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	(6,8)
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	(7,8)
(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)	(8,8)

Se divide su posición entre la razón de cambio

C siendo la razón de cambio en este caso es 2

(Aij,Bij)/c

(0.5,0.5)	(0.5,1)	(0.5,1.5)	(0.5,2)	(0.5,2.5)	(0.5,3)	(0.5,3.5)	(0.5,4)
(1,0.5)	(1,1)	(1,1.5)	(1,2)	(1,2.5)	(1,3)	(1,3.5)	(1,4)
(1.5,0.5)	(1.5,1)	(1.5,1.5)	(1.5,2)	(1.5,2.5)	(1.5,3)	(1.5,3.5)	(1.5,4)
(2,0.5)	(2,1)	(2,1.5)	(2,2)	(2,2.5)	(2,3)	(2,3.5)	(2,4)
(2.5,0.5)	(2.5,1)	(2.5,1.5)	(2.5,2)	(2.5,2.5)	(2.5,3)	(2.5,3.5)	(2.5,4)
(3,0.5)	(3,1)	(3,1.5)	(3,2)	(3,2.5)	(3,3)	(3,3.5)	(3,4)
(3.5,0.5)	(3.5,1)	(3.5,1.5)	(3.5,2)	(3.5,2.5)	(3.5,3)	(3.5,3.5)	(3.5,4)
(4,0.5)	(4,1)	(4,1.5)	(4,2)	(4,2.5)	(4,3)	(4,3.5)	(4,4)

Se redondea (Aij,Bij) al valor mayor más cercano siempre y cuando se trate de valores decimales, si es entero se conserva el valor

(1,1)	(1,1)	(1,2)	(1,2)	(1,3)	(1,3)	(1,4)	(1,4)
(1,1)	(1,1)	(1,2)	(1,2)	(1,3)	(1,3)	(1,4)	(1,4)
(2,1)	(2,1)	(2,2)	(2,2)	(2,3)	(2,3)	(2,4)	(2,4)
(2,1)	(2,1)	(2,2)	(2,2)	(2,3)	(2,3)	(2,4)	(2,4)
(3,1)	(3,1)	(3,2)	(3,2)	(3,3)	(3,3)	(3,4)	(3,4)
(3,1)	(3,1)	(3,2)	(3,2)	(3,3)	(3,3)	(3,4)	(3,4)
(4,1)	(4,1)	(4,2)	(4,2)	(4,3)	(4,3)	(4,4)	(4,4)
(4,1)	(4,1)	(4,2)	(4,2)	(4,3)	(4,3)	(4,4)	(4,4)

Se reducen los iguales dejándolos en un solo valor representativo y se le suma el valor de cambio a la posición siempre que avance en el espacio, lo cual hace el método internamente, por lo que en el código no es explícita la suma

(1,1)	(1,3)	(1,5)	(1,7)
(3,1)	(3,3)	(3,5)	(2,7)
(5,1)	(5,3)	(5,5)	(3,7)
(7,1)	(7,3)	(7,5)	(7,7)

Se le asigna el valor que le corresponde en la posición en la matriz original

220	204	198	203
203	124	128	125
194	127	136	132
203	161	156	155

y así obtenemos la imagen comprimida comprimida.

A continuación un ejemplo de una imagen y su comprimida en una escala de 9:1

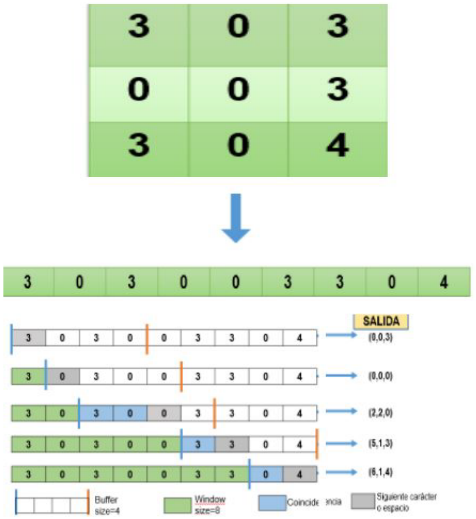
Original

Como se puede ver se comprime en pixeles.

Este algoritmo tiene las ventajas de ser rápido y sin consumir muchos recursos, y con una complejidad aceptable. También retorna resultados con una razón de compresión consistentes para cada relación de aspecto de las imágenes. Por último, permite cierto grado de libertad pues permite definir la razón de cambio al usuario, aunque por defecto está configurada a 3 para una razón de compresión aproximada a 9:1. Esto facilita adaptar el algoritmo a diferentes tipos de hardware en los que pueda correr.

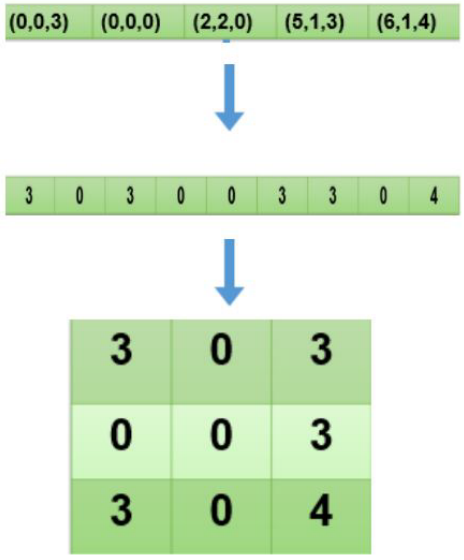
4.2.2 Algoritmo de compresión de imágenes sin pérdida

Para la compresión y descompresión de imágenes sin pérdida usamos el algoritmo LZ77. Para la compresión se debe convertir una matriz de píxeles en un arreglo; luego de tener la matriz vista como un arreglo se debe tener un tamaño de ventana y de buffer, estos tienen que tener un buen balance, es decir, si se pone un buffer muy pequeño la matriz se comprimirá muy poco y por otro lado, un buffer grande significa mayor tiempo de ejecución. Luego de tener esto, el compresor creará tuplas de la forma (p,l,c) en donde p es la cantidad de espacios que se mueve hacia atrás para encontrar una repetición, l es la cantidad de espacios que toma para la repetición y c es el valor que continua; este método consiste básicamente en encontrar repeticiones y almacenarlas en formas de tuplas que contienen toda la información necesaria.



.[17] Compresión mediante LZ77

Por otro lado, para la descompresión se toman estas tuplas que se obtienen en la compresión y se seguirán las indicaciones contenidas en la información que brindan estas tuplas; es decir, p=cuántos espacios me devuelvo hacia atrás, l= cuántos espacios tomo, c=qué carácter sigue, finalmente este arreglo se debe convertir nuevamente en matriz.



4.3 Análisis de la complejidad de los algoritmos

A través de análisis propios a mano de los algoritmos escritos, con los módulos cProfiler, time y las capacidades de graficación de Excel se confirmaron dichas complejidades para obtener las siguientes complejidades de los algoritmos.

Algoritmo	La complejidad del tiempo
Compresión	O(N*M)
Descompresión	O(N*M)

***Tabla* 1:** Complejidad temporal de los algoritmos de compresión y descompresión con perdidas de imágenes. N siendo las filas y M las columnas de la matriz más grande en el procedimiento evaluado.

Algoritmo	Complejidad de la memoria
Compresión	Depende del windows size
Descompresión	Depende del windows size

***Tabla* 2:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes.

5. RESULTADOS

En esta sección presentamos algunas métricas para evaluar los métodos de compresión.

5.1 Tiempos de ejecución

Por un lado, para la compresión con pérdidas y sin pérdidas se utilizó una selección de píxeles por Nearest Neighbor, y LZ77 respectivamente. Por otro lado, se usó LZ77 para la descompresión.

	Tiempo promedio de ejecución (s)	***Tamaño promedio del archivo (MB)***
Compresión	0.19 s	1.77 MB
Descompresión	0.08 s	0.98 MB

Tabla 3: Tiempo de ejecución de los algoritmos para diferentes imágenes en el conjunto de datos.

5.2 Consumo de memoria

Presentamos el consumo de memoria de los algoritmos de compresión y descompresión en la Tabla 7, evaluados con el módulo de Python, memory_profiler.

LZ77	***Consumo promedio de memoria (MB)***	***Tamaño promedio del archivo (MB)***
Compresión	234.1 MB	1.78 MB
Descompresión	218.98 MB	0.98 MB

Tabla 4: Consumo promedio de memoria de todas las imágenes del conjunto de datos, tanto para la compresión como para la descompresión.

5.3 Tasa de compresión

Presentamos los resultados de la tasa de compresión del algoritmo en la Tabla 8.

	Ganado sano	***Ganado enfermo***
Tasa de compresión promedio	2:1	2:1

Tabla 5: Promedio redondeado de la tasa de compresión de todas las imágenes de ganado sano y ganado enfermo.

6. DISCUSIÓN DE LOS RESULTADOS

Se observa que en los resultados que el consumo de memoria es alto pero no demasiado variable y que. Además se reconoce que la implementación de LZ77 utilizada finalmente aumenta el tamaño de los archivos, aún cuando estos se descomprimen correctamente, aunque resulten en matrices con elementos no compatibles con la descompresión bilineal. Aún así, la razón de compresión final es aceptable pues las pérdidas del escalamiento de imágenes eliminan más información de la que agrega LZ77.

6.1 Trabajos futuros

A futuro nos enfocamos a mejorar 4 cosas: la implementación de LZ77, intercambiar el escalado de imágenes por una transformación rápida de Fourier para mejorar la calidad, implementar ambos objetivos en un lenguaje compilado para disminuir los requerimientos energéticos de la compresión, y terminar el modelo de clasificación.

RECONOCIMIENTOS

Especial agradecimiento a la beca ANDI por soportar los estudios del Estudiante Jeronimo Osorio M. Y al apoyo del monitor Kevin por su asistencia.

---

[1]https://github.com/josoriom2/ST0245-001

REFERENCIAS

1.D Berckmans. 2017. General introduction to precision livestock farming. Anim. Front. 7, 1 (January 2017), 6–11.

2. Tuong Ly Le and Minh Huan Vo. 2018. Lossless Data Compression Algorithm to Save Energy in Wireless Sensor Network. Proc. 2018 4th Int. Conf. Green Technol. Sustain. Dev. GTSD 2018 (December 2018), 597–600.

3. Qin Lu, Wusheng Luo, Jidong Wang, and Bo Chen. 2008. Low-complexity and energy efficient image compression scheme for wireless sensor networks. Comput. Networks 52, 13 (September 2008), 2594–2603.

4. Parul Pandey, Mehdi Rahmati, Waheed U. Bajwa, and Dario Pompili. 2021. Real-time In-network Image Compression via Distributed Dictionary Learning. IEEE Trans. Mob. Comput. (2021).

5.Odriozola, E. Problemas sanitarios en bovinos vinculados a la intensificación ganadera. 2004.

6. Sai Ma, Qinping Yao, Takashi Masuda, Shogo Higaki, Koji Yoshioka, Shozo Arai, Seiichi Takamatsu, and Toshihiro Itoh. 2021. Development of Noncontact Body Temperature Monitoring and Prediction System for Livestock Cattle. IEEE Sens. J. 21, 7 (April 2021), 9367–9376.

7. Discrete cosine transform - Wikipedia. En.wikipedia.org, 2021. https://en.wikipedia.org/wiki/Discrete_cosine_transform.

8.https://www.spiedigitallibrary.org/ContentImages/Journals/JEIME5/23/6/061110/FigureImages/JEI_23_6_061110_f001.png

9.Fractal compression - Wikipedia. En.wikipedia.org, 2021. https://en.wikipedia.org/wiki/Fractal_compression.

10.https://upload.wikimedia.org/wikipedia/commons/8/81/Zasada_dzialania_ifs_1.png

11..Image scaling - Wikipedia. En.wikipedia.org, 2021. https://en.wikipedia.org/wiki/Image_scaling#Algorithms.

12. https://cdn.write.corbpie.com/wp-content/uploads/2021/02/video-interpolation-outcomes.jpg

13. Enhanced Compressed Wavelet - Wikipedia Es.wikipedia.org,2021. https://es.wikipedia.org/wiki/Enhanced_Compressed_Wavelet.

14. Lossless Image Compression - an overview | ScienceDirect Topics. [Fotography] https://www.sciencedirect.com/topics/computer-science/lo

15. Lossless JPEG - Wikipedia. Retrieved August 14, 2021 from https://en.wikipedia.org/wiki/Lossless_JPEG#LOCO-I_algorithm

16. An example of arithmetic coding, the source symbols are a, b, c with... | Download Scientific Diagram. Retrieved August 14, 2021 from https://www.researchgate.net/figure/An-example-of-arithmetic-coding-the-source-symbols-are-a-b-c-with-pa-0011-pb_fig2_47696414

17. Techie Delight. Huffman Coding [Fotography]. https://www.techiedelight.com/huffman-coding/.

18. Asawari Kulkarni. 2015. Gray-Scale Image Compression Techniques: A Review. Int. J. Comput. Appl. 131, 13 (2015), 975–8887.