ALGORITMOS DE COMPRESIÓN PARA LA OPTIMIZACIÓN DE PROCESOS EN LA GANADERÍA DE PRECISIÓN

John Alexander Acevedo Serna
Universidad Eafit
Colombia
jaacevedos@eafit.edu.co

Tomás Bernal Zuluaga
Universidad Eafit
Colombia
tbernalz@eafit.edu.co

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

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

RESUMEN

La existencia del ganado es primordial para los humanos, pues se ha convertido en una de nuestras principales fuentes de alimento. Por esta razón su cuidado es fundamental si queremos seguir usándolos como fuente de alimento. Con este proyecto buscamos una manera de reducir el consumo de recursos tecnológicos en las granjas que usan la GdP(Ganadería de precisión) para el cuidado de sus animales, esto por medio de un algoritmo de compresión de imágenes, que reduzca la cantidad de recursos necesarios para manejar una granja con estas tecnologías

1. INTRODUCCIÓN

La ganadería es una de las principales fuentes de alimentos para el ser humano, y en la actualidad con el número creciente exponencialmente de la población mundial, es aún más importante mejorar la efectividad de esta. En la búsqueda por ser más eficientes a la hora de realizar los procesos en ganadería, la Ganadería de Precisión (GdP) se posiciona como una buena solución, pues permite desde las tecnologías de la información y la comunicación mejorar todos estos procesos. Una de las tecnologías usadas, es el análisis de imágenes para detectar la salud de cada animal.

El problema del uso de estas tecnologías para la ganadería es que sin importar lo mucho que faciliten las tareas, también van a requerir un gran gasto para los ganaderos, pues por lo general en estas granjas no se dispone de tan fácil acceso a estas tecnologías ni las conexiones a internet son muy buenas.

Por lo anterior se presentó como una solución al problema el uso de algoritmos de comprensión para imágenes, pues al reducir el tamaño de las imágenes captadas para detectar la salud de los animales, el tiempo de carga y de revisión de estas fotos se vería enormemente reducido y el proceso sería más sencillo, permitiendo así que los ganaderos puedan hacer uso de estas tecnologías sin inconvenientes.

1.1. Problema

El problema al cual le plantamos solución tiene como base la ganadería de precisión, la cual se encarga de mejorar la eficiencia productiva por medio de recursos e innovaciones tecnológicas.

Una problemática son los recursos tecnológicos limitados que se encuentran en las instalaciones ganaderas, esto impide que tanto el tiempo de procesamiento, como los recursos de las máquinas no soporten imágenes de gran tamaño.

Es por eso que la compresión de imágenes del ganado para clasificar la salud animal y su buena implementación traería consigo la optimización tecnológica necesaria para seguir avanzando en la ganadería de precisión.

1.2 Solución

En este trabajo, utilizamos una red neuronal convolucional para clasificar la salud animal, en el ganado vacuno, en el contexto de la ganadería de precisión (GdP). Un problema común en la GdP es que la infraestructura de la red es muy limitada, por lo que se requiere la compresión de los datos.

Como una solución al problema, se plantea el uso de algoritmos de comprensión para imágenes, pues reduciendo el espacio que ocupan en memoria las imágenes usadas para la GdP, se verían enormemente reducidos los tiempos de carga y su uso sería mucho más sencillo para los ganaderos.

Nosotros planteamos un algoritmo que reciba las imágenes en formatos .csv, y encuentre todos los valores que son similares, para así proporcionar un promedio y un número de veces que aparecen estos datos. Con esto se guardan en objetos los datos similares o iguales y sus respectivas repeticiones, reduciendo la cantidad de valores presentes en el archivo .csv inicial y por ende su tamaño en memoria. Todo esto presentando perdidas muy poco notables en la imagen final.

2. TRABAJOS RELACIONADOS

2.1 Desarrollo e implementación de un dispositivo de adquisición y almacenamiento de sonidos para ganadería de precisión

Hoy por hoy siguen existiendo una serie de dificultades que ponen contra las cuerdas a los ganaderos a la hora de mejorar y optimizar la producción. Una dificultad que sigue siendo la falta de información certera sobre la alimentación del ganado. Es por eso que el desarrollo e implementación de un dispositivo de adquisición y almacenamiento de sonidos para ganadería de precisión mejoraría y optimizaría la producción. Para hacer posible esto se planteó un sistema que contiene 3 módulos, el primero se centra en la adquisición y limpieza del sonido emitido por el animal, el segundo se enfoca en la compresión del sonido, su organización y almacenamiento, y el tercer módulo se encamina en la administración de la energía para su funcionamiento. Para implementar el sistema propuesto se utilizó un algoritmo de compresión de audio, y el elegido fue CELT, que es un algoritmo de compresión con perdida que tiene como principal característica tener una complejidad lo suficientemente baja para procesadores integrados de baja gama. Fue escogido tanto por lo anterior como porque no se necesitan muchos recursos de hardware para ser implementado y su tiempo de latencia es bajo [1].

2.2 Visual localization and individual identification of Holstein Friesian Cattle via Deep Learning

Desde los inicios de la ganadería, la raza Holstein ha estado presente en el podio de los mayores productores de leche a nivel global, es por eso que se debe destacar la ganadería de precisión en esta raza, ya que, tanto su constante importación como su exportación se ven desaceleradas por la dificultad al identificar cada res del ganado, ya que las técnicas más frecuentes para la identificación de esta raza han sido la marcación con tatuajes y la marcación en las orejas. Ambas técnicas desaceleran el proceso de identificación porque su complejidad es alta al realizarse sobre grandes cantidades de ganado. Además, son métodos intrusivos que afectan a mal directamente al bienestar del rebaño. Es debido a esto que se planteó un sistema que se encarga de identificar cada miembro del ganado de manera automática, veloz y de manera no intrusiva. Para esto se empleó un sistema/proceso que comienza con el Region Based Convolutional Neural Networks (RCNN), que se encarga de producir imágenes de regiones de interés (vacas) (ROLS) o en forma de cuadros. Para este paso el estudio se centró en dos algoritmos, el ScaleInvariant feature Transform (SIFT) para reconocer objetos (vacas) y su movimiento; y el Speeded-Up Robust features (SURF) que es un complemento del anterior al poder extraer información específica y detallada de una imagen. Con esto ya se genera la localización de la vaca y una especie de ID según los rasgos que se identifiquen en el ROLS. Después se pasa al proceso de clasificación individual que se ejecuta con el algoritmo de seguimiento, para finalizar en el long short term memory (LSTM) que se utiliza debido a su capacidad de "recordar" estados previos y utilizar esta información para deliberar lo siguiente en el proceso [2].

2.3 Cloud services integration for farm animals behavior studies based on smartphones as activity sensors

En este proyecto, sus autores buscaban la forma de reunir de la mejor manera los datos de comportamiento de animales en una granja haciendo uso de la ganadería de precisión. Esto lo hicieron ya que estas estadísticas sirven para llevar un control exacto de los animales, y permite además encontrar alternativas de solución para ciertos problemas que se descubran durante el análisis y procesamiento de los datos recolectados.

Para este estudio se usaron celulares IPhone 4s y 5s, poniendo a prueba sus capacidades para recolectar los datos y la duración de sus baterías al realizar estas tareas. También se pensó en la forma de reducir el consumo de batería y mejorar la eficiencia del procesamiento de datos por medio del Edge Computing, aunque al final encontraron que esto era innecesario pues no mejoraba en mucho la autonomía de los celulares ni la velocidad de traslado a la nube. Para la compresión de datos usaron varios algoritmos que redujeron la cantidad de decimales presentes en los datos, que era un número de 6 y buscaban bajarlo a 4 o menos, y que borrara los datos redundantes (datos que son muy parecidos entre sí), por ejemplo, en la velocidad de movimiento de los animales donde casi todo el tiempo se mantenía constante [3]. Para los datos en la nube buscaron el uso de una estructura lambda que permitiera una mejor comunicación entre los distintos equipos conectados a la base de datos, además de que este modelo ofrece muchas posibilidades a la hora de implementar funciones adicionales, dejando cargar imágenes y datos externos de otras nubes sin ningún inconveniente.

Los investigadores terminaron concluyendo que era viable usar estos celulares como dispositivos de toma de datos en bovino por su bajo costo y buen desempeño; también concluyeron que era mejor comprimir los archivos individualmente pues de esta forma la tasa de compresión aumentaba en algunos casos hasta el doble, pues al comprimir en grupos solo se

alcanzaba como máximo un 43.5% [3]. Con todo esto se comprobó que era posible hacer uso de la ganadería de precisión sin necesidad de una infraestructura tecnológica muy avanzada.

2.4 An Animal Welfare Platform for Extensive Livestock Production Systems

En este proyecto se planteó la importancia de la ganadería en la actualidad y cómo por esta razón es importante tener un buen control sobre el bienestar en los animales. El problema que surgió era la dificultad para tener monitoreados a los animales en granjas muy extensas, donde el costo para una infraestructura con sensores por zonas fuera muy elevado. Por ello, plantearon una solución haciendo uso de dispositivos en forma de collar que se colocarían en el cuello de los animales. Estos dispositivos llevarían el control de los movimientos del animal para velar por su seguridad. Todo esto sería monitoreado desde una aplicación desarrollada para los celulares de los granjeros.

Durante el proyecto concluyeron que también era necesario poseer dispositivos para edge computing, ya que enviar todos los datos directamente a la nube para después ser procesados allí no era eficiente; por lo que en estos dispositivos se procesaría y se almacenaría de manera organizada la información para después ser enviada a la nube y allí se ejecutarían otros procesos para mejorar el sistema por medio de Deep machine learning y después se enviarían los datos a la aplicación de celular creada.[4]

3.2 Alternativas de compresión de imágenes con pérdida

En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes con pérdida.

3.2.1 Compresión fractal

La palabra fractal es un término que se define como un patrón geométrico que se repite en varias escalas. Por tanto, como su nombre lo indica, este método de compresión busca la semejanza entre distintas figuras que se forman en una imagen; algo que se considera muy común en texturas naturales, pues por lo general sus partes tienen mucha semejanza.

Como se deben buscar todas las posibles similitudes presentes en una imagen, este método tiende a ser muy demorado, aunque por otro lado la decodificación de la imagen es bastante rápida. Por esto, el verdadero problema para el algoritmo es la complejidad y la cantidad de colores que maneja la imagen que se busca comprimir.

Como vemos en la figura 1, se busca la semejanza entre dos segmentos de una misma imagen para que después solo se guarde uno de ellos. Al final la imagen se formará como una especie de rompecabezas, donde se tienen las distintas posiciones que debe ocupar un mismo recorte.

Figura 1: ejemplo Compresión fractal

3.2.2 JPEG

Normalmente las imágenes las percibimos como si tuvieran tres canales de color, uno verde, otro rojo y otro azul. Este algoritmo de compresión transforma la manera de representar la imagen a una en la que tenemos dos canales de color y uno de brillo, reduciendo así la cantidad de colores que debe usar. Además, reduce la calidad de la imagen agregando colores por bloques más grandes, ósea que en vez de pintar de un color cada píxel, pintaría dos píxeles a la vez como vemos en el ejemplo de la figura 2. En el ejemplo se puede observar que esta compresión hace que se vea borrosa la imagen pues pierde definición al reducir la cantidad de píxeles.

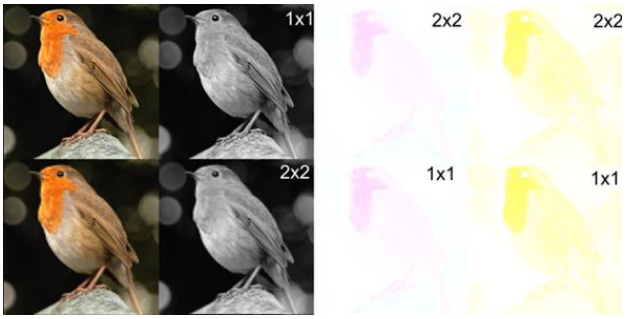

Figura 2: ejemplo JPEG

3.2.3 Seam Carving

El Seam Carving primero revisa cuales son las partes con la menor relevancia en la imagen y por lo tanto las que deberán ser eliminadas o añadidas en el proceso final, aunque

también permite la elección manual de las áreas que no deben ser modificadas debido a su importancia.

En la figura 3 vemos cómo funciona el proceso. Se elimina todo el espacio que hay entre el castillo y la persona pues es un espacio vacío que no le aporta nada a la imagen.

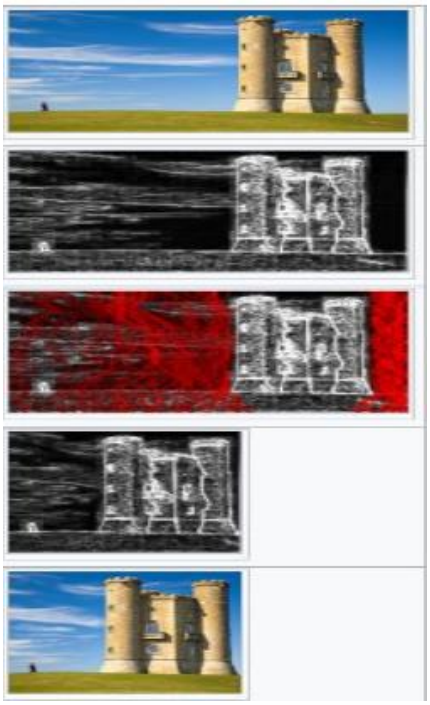

Figura 3: Seam Carving

3.2.4 EZW (Embedded Zerotree Wavelet)

Fue propuesto por Shapiro en 1993. Este método explota las propiedades aportadas por la DWT (Discrete Wavelet Transform), estas propiedades son un gran porcentaje de coeficientes wavelets próximos a cero y la agrupación de la energía de la imagen. La implementación de este método se realiza por medio del algoritmo de incrustación. Este algoritmo cuantifica los wavelets en pasos dados por potencias de dos, reduciéndose progresivamente en iteraciones.

Primero se calcula un umbral de partida, y con este se construye el Zerotree, que se basa en la hipótesis de que si un wavelet es insignificante con respecto a este umbral, todos los wavelets en ese rango lo son también, por lo que se agrupa una gran cantidad de datos.

Figura 4: Recorrido que realiza el EZW

3.3 Alternativas de compresión de imágenes sin pérdida

En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes sin pérdida.

3.3.1 Run-length encoding (RLE):

Este método de compresión se basa en la repetición de elementos consecutivos. Para imágenes es útil solo si se presentan colores uniformes, pues si no hay pixeles repetidos la compresión terminara siendo negativa. Un ejemplo con caracteres para representar esto sería con la cadena “XXXXDDDDDD” cuando está comprimida da como resultado "4X6D", reduciendo así el espacio que ocupa en la memoria. En cambio, la cadena "SUBMARINO" daría como resultado “1S1U1B1M1A1R1I1N1O”, haciendo una cadena más larga que la original, evidentemente no beneficiaria usar este método de compresión en casos como este.

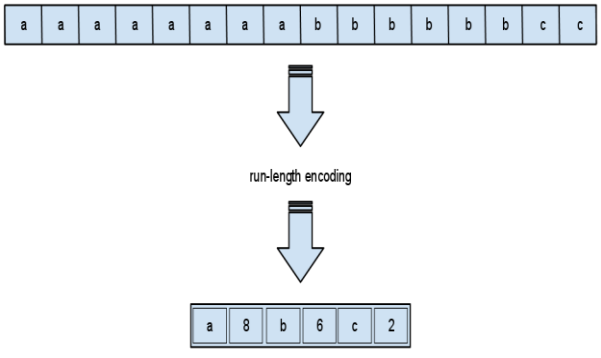

Figura 5: Ejemplo Run-length encoding (RLE)

3.3.2 Lempel ziv (LZW):

Este método consiste en realizar un análisis detallado de una cadena de un alfabeto, en el que se asigna a cada cadena un código único de palabras con longitud fija LZ. Las cadenas se seleccionan en un formato tal que los elementos con mayor frecuencia se agrupan en cadenas más largas que los elementos con menor frecuencia, los cuales son agrupados en cadenas cortas. Cuando se está comprimiendo una secuencia de bits, se sigue un proceso en el que si los caracteres coinciden con alguna sucesión anterior, se intercambia por la tripleta (m,n,s) donde m es el lugar hacia atrás donde se inicia la sucesión , n la longitud de la secuencia anteriormente encontrada y s el siguiente carácter de la cadena comprimida.

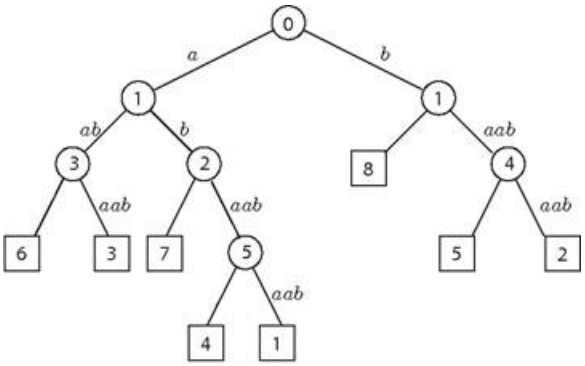

Fig. 1. The suffix tree of $x = abaabaab$.

Figura 6: Ejemplo Lempel ziv (LZW)

3.3.3 LZ77

Este algoritmo de compresión sin perdida está basado en diccionario y su funcionamiento de salida se centra en 3 formas: la primera son los literales, estos son bytes sin comprimir; la segunda son las palabras clave, que son una especie de bytes con información para saber si el dato a analizar es un comprimido; y el último son las banderas, las cuales indican si el dato a analizar es un literal o una palabra clave. Estas 3 formas se emplean al recorrer una cadena.

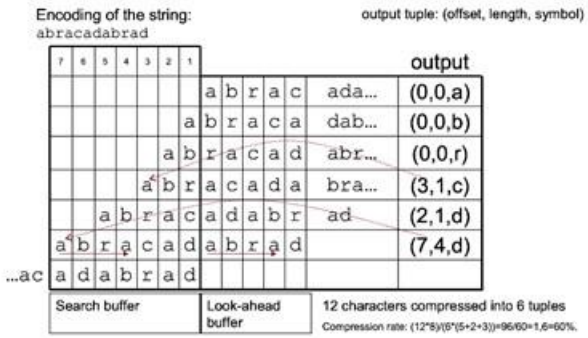

Figura 7: Ejemplo LZ77

3.3.4 Codification de Huffman

Esta técnica se fundamenta en escoger la representación de cada símbolo, de tal manera que se asigna la cadena de bits más corta a los datos que más aparezcan y las cadenas más largas a los que aparecen con menor frecuencia.

El algoritmo se basa en un árbol binario de nodos que va de abajo hacia arriba, de más a menos específico.

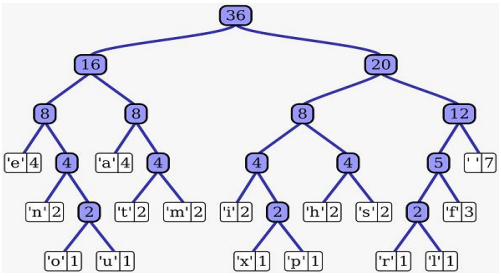

Figura 8: Ejemplo Codificación de Huffman

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

Las estructuras de datos que empleamos para la compression de las imagenes fueron matrices y arreglos, en los que se guardabamos los datos en formato .csv despues del proceso de compresión. Estas estructuras Tambien nos sirvieron durante la ejecución de los algoritmos, pues facilitaban el acceso a los datos

El algoritmo de compression recibe el archive como imagen en formato .jpg, despues en el algoritmo de compression se pasa a una matriz en .csv o a un arreglo en el caso del LZ77

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.

Usamos un algoritmo que reciba las imágenes en formatos .jpg y por medio de una interpolación de imágenes haciendo uso del algoritmo del vecino más cercano, halle los valores similares alrededor de cada píxel y guarde el valor promedio en un solo píxel de la nueva imagen, para que así pueda reducir la imagen final a un 30% del tamaño original. Después de esto se introducirá la imagen comprimida al algoritmo LZ77 que la comprimirá sin perdidas, haciendo que las cadenas de pixeles se vean disminuidas de tal forma que se reemplacen las largas ocurrencias repetidas, dejando al final una marca determinada donde se encuentra la compresión total.

4.2.1 Algoritmo de compresión de imágenes con pérdida

Nuestro algoritmo de compression con perdidas hace uso del algoritmo del vecino más cercano, que se ancarga de revisar los valores que hay en un radio alrededor de cada valor, para asi despues hallar un promedio que se introducirá en una nueva estructura reemplazando los valores anteriores. Para escalado de imágenes es muy usado, pues permite tanto reducir como aumentar el tamaño de una imagen, y comparado con otros algoritmos de interpolación de imágenes, su distorsión no es tan notable.

El algoritmo recibe una imagen en formato .jpg y dentro del algoritmo se asigna la escala a la que se quiere reducer la imagen, todo estos se hace por medio de la librería OpenCV de python, que tiene una función para implementar de manera sencilla este algoritmo. Despues de comprimir la imagen se puede guardar en una matriz con formato .csv, y para descomprimirla se ingresa esta imagen en formato.jpg y se hace el proceso contrario a la compression, se escala la imagen para agrandarse, volviendo así a su tamaño original solo que con unas breves distorsiones debido a la compresión.

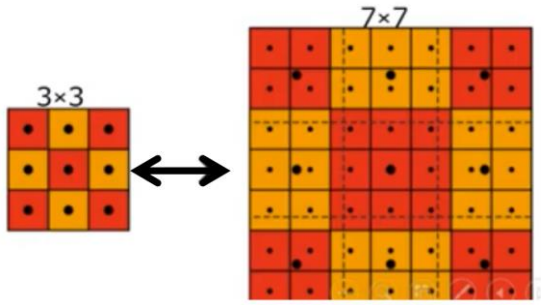

Figura 9: Representación gráfica del funcionamiento de la interpolación de imágenes hacienda uso del algortimo del vecino más cercano

4.2.2 Algoritmo de compresión de imágenes sin pérdida

El algoritmo que empleamos para la compresión sin pérdida fu el lz77, este se encarga de encontrar cualquier coincidencia nueva en el búfer de búsqueda. Para esto halla la coincidencia más larga de una cadena y luego genera una triple marca, en donde la primera representa el número deposiciones que necesitaríamos mover hacia atrás para encontrar el inicio de la cadena que coincide, la segunda representa la longitud después del encuentro, y la tercer representa el carácter que se encuentra después de la cadena que coincide. Es por esto que la función de compresión recibe una cadena de pixeles que representan la imagen a comprimir. Al final de la descompresión se debe tener otra vez la cadena de pixeles exactamente igual la inicial, para esto se toma cada tripleta, primero se busca en el buffer y luego se escribe en la cadena final.

4.3 Análisis de la complejidad de los algoritmos

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

Para la compression con pérdidas, la complejidad en tiempo es tan solo de O(N*M), pues se tiene que leer todos los píxeles de la imagen una vez y agregar los valores comprimidos a una nueva imagen. Y para la descompresión es lo mismo. En el caso del algoritmo LZ77 la complejidad es de $O(N^2*M^2)$

Algoritmo	**La complejidad del tiempo**
Compresión	$O(N^2*M^2)$
Descompresión	$O(N^2*M^2)$

***Tabla* 2:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. (N son el número de filas de píxeles por imagen, y M el número de columnas)

Algoritmo	**Complejidad de la memoria**
Compresión	O(N*M)
Descompresión	O(N*M)

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. (N son el número de filas de píxeles por imagen, y M el número de columnas)

4.4 Criterios de diseño del algoritmo

El algoritmo fue hecho para que todos los datos se almacenaran de la manera más ordenada y simple posible, para facilitar su acceso. Se escogió el escalado de imágenes manteniendo la relación de aspectos de la imagen y hacienda uso del algoritmo del vecino más cercano, ya que, entre los demás algoritmos de interpolación de imágenes, este fue el que mostró mayor desempeño en cuanto a mantener la calidad de la imagen al descomprimir. Por otro lado, el algoritmo de compresión sin pérdidas LZ77 demostró la capacidad de comprimir y descomprimir imágenes utilizando cantidades racionales de tiempo y memoria, además, la principal ventaja que demostró en las pruebas fue su rápida descompresión.

5. RESULTADOS

5.2 Tiempos de ejecución

En lo que sigue explicamos la relación entre el tiempo promedio de ejecución y el tamaño promedio de las imágenes del conjunto de datos completo, en la Tabla 6.

	Tiempo promedio de ejecución (s)	***Tamaño promedio del archivo (MB)***
Compresión	83.4 s	1.2MB
Descompresión	8.2 s	0.26 MB

Tabla 6: Tiempo de ejecución de los algoritmos escalado de imagenes con el vecino más cercano, y LZ77 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.

	Consumo promedio de memoria (MB)	***Tamaño promedio del archivo (MB)***
Compresión	1.8MB	1.2MB
Descompresión	0.4 MB	0.26 MB

Tabla 7: 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	1:5	1:6

Tabla 8: Promedio redondeado de la tasa de compresión de todas las imágenes de ganado sano y ganado enfermo.

6. DISCUSIÓN DE LOS RESULTADOS

Los resultados obtenidos fueron los esperados, despues de usar los dos algoritmos de compresión. El tamaño de la

imagen se vió enormemente reducido, y además se conservaron sus caracteristicas, permitiendo que fuese facilmente identificable al verla. No obstante, se podría lograr una mayor tasa de compresión con una implementación más personalizada adecuandose resoluciones y los datos de cada imagen. En general los algoritmos funcionarían para cualquier imagen, sin importer que no sea ganado. Los tiempos de ejecucución no fueron extremadamente rapidos, pero teniendo en cuenta todo el trabajo que se debía hacer, especialmente en la ejecución del LZ77, fueron bastante buenos, por esto estamos satisfechos con la elección del algoritmo.

6.1 Trabajos futuros

Nos gustaría poder organizar las imágenes y las matrices de manera más organizada en carpetas por medio de nuestro programa, además mejorar la efectividad del escalado de imágenes realizando todos los ajustes manualmente dependiendo del tipo de imagen que se quiera comprimir, para que así los resultados finales sean mejores. Sería interesante ver como a futuro, aparte de lograr mejorar con lo que se mencionó anteriormente, se evitará aun más la distorsión al comprimir la imagen por medio del uso de ondaletas; todo esto con un algoritmo que guarde las ondaletas de las zonas que necesiten mayor claridad, para mantener lo más cercana en detalle la imagen comprimida a la original.

RECONOCIMIENTOS

Agradecemos la asistencia con las ideas de implementación para los algoritmos brindadas por compañeros del curso y por nuestros padres, que gracias a esto ideamos formas eficientes de implementar estos algoritmos de compresión para imágenes.

REFERENCIAS

1. J. Chelotti, C. Arrasin, S. Vanrell, H. Rufiner, L. Giovanini. *Desarrollo e implementación de un dispositivo de adquisición y almacenamiento de sonidos para ganadería de precisión*. 6º Congreso Argentino de AgroInformática, CAI, 2014.

2. W. Andrew, C. GreatWood, T. Burghardt. *Visual localization and individual identification of Holstein Friesian Cattle via Deep Learning*. Bristol 2018.

3. Olivier Debauche, Said Mahmoudi, Andriamasinoro Herinaina, Pierre Manneback, Jerome Bindelle, Frederic Lebau. Journal of Ambient Intelligence and Humanized Computing. 2019. Cloud Services Integration for Farm Animals Behavior Studies Based on Smartphones as Activity Sensors. (May. 2018). 4651-4662.

4. Vasileios Doulgerakis, Dimitrios Kalyvas, Enkelada Bocaj, Christos Giannousis, Michalis Feidakis, George Laliotis, Charalampos Patriakis. An Animal Welfare Platform for Extensive Livestock Production Systems.