Algoritmos de compresión para la optimización del consumo de baterías en ganadería de precisión

[1]John Esteban Castro Ramirez [1]Carlos Gustavo Vélez Manco, [1]Johan Esteban Mesa Vanegas, [1]Simón Marín, [1]Mauricio Toro
[1]Universidad Eafit
[1]Colombia
[1]jecastror@eafit.edu.co, [1]cgvelezm@eafit.edu.co,[1]jemesav@eafit.edu.co, [1]smaring1@eafit.edu.co,[1]mtorobe@eafit.edu.co

Resumen

La existencia del ganado es primordial para los humanos, ya que proporciona el 33 % de nuestra dieta. Por tanto, es muy habitual encontrarse con el uso de herramientas tecnológicas que optimicen los procesos que se desarrollan con estos animales, como por ejemplo la ganadería de precisión, la cual monitoriza y automatiza continuamente la salud animal; sin embargo, hacen falta algunos ajustes, pensando en el público que suele usar este recurso, por lo que se hacen necesarios programas que minimicen diferentes recursos que en muchas ocasiones son escasos. Existen gran variedad de dificultades enlazadas con la anteriormente descrita, las cuales se analizarán, para encontrar una solución más completa y/o robusta.

En este trabajo, realizamos una compresión y descompresión de imágenes con el fin de generar un mayor ahorro de energía y tiempo. El algoritmo usado es una combinación de compresión de imágenes con pérdida y sin pérdida, en este caso del escalamiento mediante el vecino más cercano y el LZ77 respectivamente; obtuvimos unas buenas tasas de compresión, en la mayoría a la octava parte de la imagen y a la vez no perdimos muchos datos durante la compresión logrando un buen balance entre consumo de energía y tiempo.

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

1. Introducción

La historia de la ganadería en América, se remonta a la época de la conquista española, el ganado llegó con los españoles que lo usaban para su alimentación ya que los distintos pueblos indígenas no tenían animales domésticos para el consumo; en el siglo XX esta industria tomó gran importancia para la sociedad, debido al aumento demográfico. [1]

Debido al incremento constante del consumo de los productos derivados del ganado, han surgido necesidades para optimizar las labores de los trabajadores de este sector y garantizar elementos de calidad; como respuesta a este fenómeno, aparecen distintas herramientas tecnológicas, como la ganadería de precisión, que busca monitorear y automatizar los diferentes procesos de manejo de estos animales, este componente ha sido de gran ayuda, pero aún existen ciertos problemas que impiden el acceso a estos instrumentos para todo el público involucrado, por lo que se hacen esenciales ciertas modificaciones que permitan llegar a una cantidad masiva de clientes.

Para darle solución a este problema, se deben de tener en cuenta diferentes factores en cuanto a recursos, sobre todo tiempo y memoria. Cumpliendo así el objetivo principal de optimizar recursos.

1.1. Problema

El horizonte de este trabajo es desplegar un algoritmo, haciendo uso de estructuras de datos, que permite comprimir y descomprimir las imágenes del ganado, generando así una mayor optimización del tiempo y memoria requeridos tanto para procesar la información como para clasificar las reses según su salud.

La resolución de esta problemática, sería de gran ayuda para empresas, ganaderos y zootecnistas que usan la ganadería de precisión, ya que les permitiría usar el algoritmo incluso con una conexión a internet inestable, que es lo que comúnmente sucede en los sitios en los cuales se encuentran alojados estos animales; y así favorecer la labor de estos entes, asegurando a la población un consumo de leche y carne de calidad.

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.

En primera instancia, desarrollamos un algoritmo para la compresión y descompresión con pérdidas: 'escalamiento de la imagen por medio del vecino más cercano'; ya que teniendo en cuenta el tipo de imágenes que vamos a manejar, lo más factible es comprimir la imagen por medio de la reducción de su tamaño y la pérdida de un poco de su calidad comparado con la remoción de elementos que utiliza el tallado o el gran costo que puede significar la compresión fractal, para efectos posteriores del uso del algoritmo para la clasificación de salud animal, es más viable tener la imagen estilo pixel art que una imagen baja de calidad.

Luego, es evidente que esta compresión y descompresión con pérdidas puede generar una reducción considerable de la precisión del algoritmo de clasificación de salud animal, por lo que se hizo necesario implementar un segundo algoritmo para la compresión sin pérdidas.

Para la compresión y descompresión de imágenes sin pérdida aplicamos el algoritmo LZ77, teniendo en cuenta su complejidad de tiempo y memoria sobre las otras técnicas de compresión sin pérdida que podrían tener un mejor desempeño en memoria sobre el LZ77 pero su complejidad en tiempo es mayor y por ende, teniendo en cuenta que para este proyecto es más priorizable el tiempo que la memoria, optamos por el LZ77.

Así, para optimizar el proceso, a las imágenes primeramente se les aplica el escalado y luego se les aplica el LZ77 y para la descompresión se les aplica primeramente el LZ77 y luego el escalado mediante el vecino más cercano.

1.3. Estructura del artículo

En lo que sigue, en la Sección 2, presentamos trabajos relacionados con el problema. Más adelante, en la Sección 3, presentamos los conjuntos de datos y los métodos utilizados en esta investigación. En la Sección 4, presentamos el diseño del algoritmo. Después, en la Sección 5, presentamos los resultados. Finalmente, en la Sección 6, discutimos los resultados y proponemos algunas direcciones de trabajo futuras.

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

El ganado Holstein Friesian es el mayor productor de leche tipo bovina y de allí su importancia económica, por lo que su identificación es importante para la exportación y demanda de los clientes, este proceso se suele realizar con distintas marcas como etiquetas en sus orejas, tatuajes, entre otros, que claramente son cuestionables. Por esto, este trabajo busca implementar la

automatización de reconocimiento a través de diferentes patrones en su pelaje, lo cual, contribuye a una granja más eficaz y al bienestar del animal.

En primer lugar, se encargaron de la identificación del ganado, para esto, se dieron a conocer los 3 tipos de identificación de ganado que existen: patrones de bozal o Cattle muzzle patterns, y escaneos faciales, de retina o de cuerpo y escaneos de piel, centrándose en este último. En el cual, se pudo realizar a través de algoritmos como Scale-Invariant feature Transform (SIFT) y Speeded-Up Robust features (SURF) a través de la captura de fotos de 3840x2160 pixeles tomadas a 5 metros del suelo. [2]

Gracias al Regional Convolutional Neural Networks (RCNN), se logra seleccionar un objeto, en este caso un animal, se determina su localización y se genera un ID buscando los patrones que hay en el dorso del animal. Toda esta información queda guardada en forma de secuencia, usando el Long short-Term Memory (LSTM), que organiza toda la información que recopila dependiendo de la cantidad de ganado que se tenga en la granja. A continuación, se presentan los resultados obtenidos, luego de varias validaciones.

Task	mAP (%)		
	Fold 1	Fold 2	Average
Detection & Localisation	99.02	99.59	**99.3**

Figura 1: Precisión en la detección de especies

Como se puede observar, en promedio la precisión es de 99,3 %; demasiado buena.

2.2. Ganadería de precisión en vacuno de carne

La universidad de Zaragoza realiza este trabajo con el fin de describir una gran cantidad de datos y características acerca del ganado, usando parámetros biométricos.

En principio, se seleccionaron cerca de 15.000 referentes de la utilización de GdP, y gracias a esto, el trabajo tuvo por objetivo principal analizar el vacunado de carne con la ayuda de herramientas tecnológicas que optimicen los procesos necesarios para esta tarea.

La identificación electrónica de estos animales se clasifica en dos tipos; identificación por radio de frecuencia (RFID), la cual genera una captura automática de datos e identifica animales, objetos, y hasta personas por medio de etiquetas que poseen un único número de ID que luego son puestos en objetos como placas para orejas. Una debilidad de este método es que no funciona tan bien para granjas con muy pocos animales, o cuando existen grandes cantidades de animales muy cercanos entre sí. El otro tipo de identificación es la que se basa en parámetros biométricos o morfológicos, gracias al análisis de imágenes por medio del uso de algoritmos, por ejemplo para los humanos, se tienen datos como huellas dactilares, reconocimiento de cara e iris, entre otros, y para los bovinos se tiene también varias características como impresión de hocico, imágenes faciales y reconocimiento de iris, los cuales son procedimientos que no generarán ningún tipo de dolor o molestia en los animales, porque por ejemplos los chips pueden ser molestos para ellos.

Se realizó un estudio muy detallado acerca de varios factores como reproducción del ganado, peso de los animales, vallado virtual, bienestar del animal, alimentación, tiempo de rumia, medio ambiente, etc. Pero para este tipo de análisis es de vital importancia el uso de las dimensiones corporales.

Junto con la condición corporal, estos dos factores hablan mucho acerca del animal que se está analizando, por tanto, es necesario tenerlos en cuenta a la hora de mejorar tanto la productividad del animal como la calidad del producto que estos nos brindan, ya sea carne o leche. Por otro lado, gracias a la recopilación de estos factores, se pueden optimizar procesos tales como tiempo y esfuerzo por parte humana, y niveles de dolor y estrés por la parte bovina. Todo esto, se logra gracias al sensor LIDAR o Light Detection and Rangin que obtuvo las siguientes mediciones: Altura de la cruz, profundidad del pecho, altura dorsal, longitud de cuerpo y altura de la grupa. [3]

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

En este trabajo, se utilizaron celulares, en particular Iphone 4s y 5s, como sensores para obtener ciertas medidas acerca del ganado.

Muchas características tanto de animales como de humanos, se pueden medir gracias al uso de celulares, por ejemplo, micrófono, sensores de presión, electromiografía, locación, acelerómetros, entre otros. Por ejemplo, el uso de un chip GPS y un acelerómetro en el cuello de un bovino genera un cambio aproximadamente del 90 % en el comportamiento del animal [**4**]. Los celulares poseen unidades de medida inerciales, las cuales reconocen cierto tipo de señales que pueden medir, por lo que una gran ventaja de esto es que no será necesario el uso de hardware adicional ni de herramientas que midan sólo un tipo de señal.

Se concluyó que ambos celulares poseen una precisión muy parecida, con errores entre sí de aproximadamente el 0,1 %.

2.4. Programa de bienestar animal para grandes sistemas ganaderos

El monitoreo de la salud animal en las granjas es esencial, ya que a través de este se puede obtener información importante y prevenir algunas enfermedades. [**5**]

Esto se realiza a través de sensores en forma de collar, colocados en los animales; los cuales funcionan para recolectar datos en tiempo real como la geolocalización y su actividad o bienestar, que luego se cargan a la nube y se puede acceder a ellos a través de una aplicación móvil. Para optimizar, sobre todo la batería, y asegurar la transmisión de gran cantidad de datos desde los collares hasta los dispositivos Edge para luego ser montados en la nube diariamente, se utilizan 2 canales de comunicación, el primero con un pequeño rango de comunicación, altas velocidades de transmisión de datos y gran consumo de energía, en tanto que el segundo comprende un rango más amplio, menor consumo de energía y menor velocidad de transmisión; evidentemente la comunicación se dará más a través del segundo canal para optimizar energía.

Con este servicio los ganaderos tienen la capacidad de conocer los movimientos de sus animales y estar informados; a través de los datos cargados en la nube se pueden obtener múltiples informaciones y estadísticas que les ayudan a tomar decisiones más acertadas en cuanto al futuro de sus animales y así obtener una mayor utilidad y/o beneficio.

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 çow". Para el ganado enfermo, la cadena de búsqueda era çow + 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://n9.cl/wien8

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://n9.cl/npt3b

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. Liquid rescaling

Es un algoritmo para el "re-tallado" de imágenes desarrollado por Shai Avidan y Ariel Shamir, consiste en establecer una serie de costuras o caminos de menor importancia, una vez aplicado se pueden agregar o eliminar, lo que amplía o reduce el tamaño de la imagen respectivamente.

Permite definir de manera manual las áreas en las que no se desea que se modifiquen los píxeles y presenta la habilidad de remover objetos completos de las fotografías

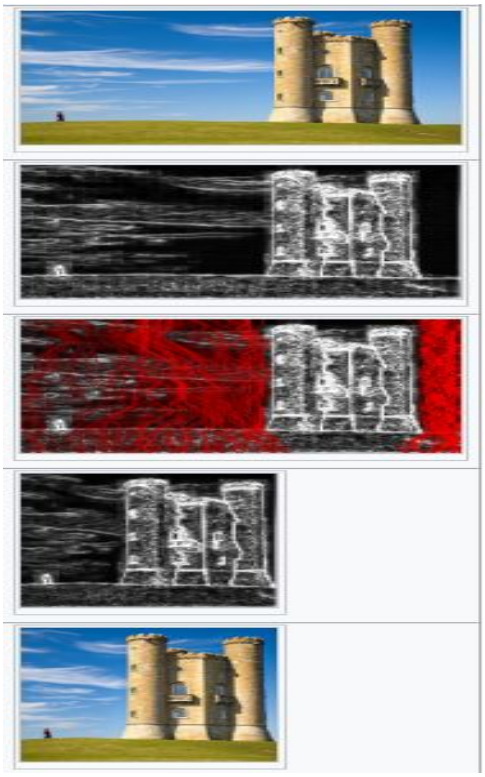

Figura 2: Proceso del Algoritmo Liquid Rescalid

3.2.2. Escalado de imágenes

Se refiere al ajuste de tamaño de una imagen digital, que en el medio de tecnología de video se conoce como "upscaling"; al momento de escalar un vector gráfico, los elementos primitivos que componen la imagen se pueden modificar usando transformaciones geométricas sin pérdida de la calidad de imagen, cuando se escalan gráficos ráster se debe generar otra imagen con un mayor o menor número de píxeles, en caso de que decrezcan (scaling down) resulta con una reducción de la calidad de imagen.

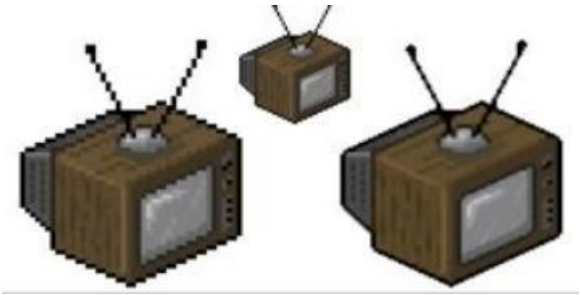

Figura 3: Ejemplo televisor a escala

Desde el punto de vista del procesamiento de señal digital, la escala de los gráficos ráster es un ejemplo bidimensional de conversión de frecuencia de muestreo, la conversión de una señal discreta de una frecuencia de muestreo (en este caso la frecuencia de muestreo local) a otra.

3.2.3. Compresión fractal

La palabra fractal es un término reciente propuesto en el año de 1975, el cual se define como un patrón geométrico que se repite en varias escalas.

Por tanto, como su nombre lo indica, la compresión fractal es un método de compresión basado en fractales, este es el más apropiado para imágenes o texturas naturales, porque ciertas partes de la imagen se parecen a otras.

Como se deben buscar similitudes propias de la imagen, este método puede llegar a ser bastante costoso, aunque, por otro lado, la decodificación de la imagen se hace de una manera bastante rápida. Por tanto, la eficiencia del algoritmo va a depender mucho en la complejidad de la imagen y

la profundidad del color.

Una de las principales características de la compresión fractal, es que las imágenes se vuelven independientes de la resolución (escalado fractal), y esto hace posible, que la imagen se pueda pasar una mejor resolución (interpolación fractal).

Figura 4: Ejemplo de compresión fractal

3.2.4. Compresión JPEG

Este algoritmo, pierde información y esto se refleja en una disminución de la calidad de la imagen; pero tal pérdida no es visible para el ojo humano. El funcionamiento del algoritmo se explica a continuación: se inicia con una imagen en donde cada píxel tiene 3 canales de color (rojo, azul y verde) y lo primero que hace el algoritmo es transformar estos 3 canales en 2 de color y uno de brillo; luego, a los canales de color les baja la resolución y allí es donde se generan las perdidas ya que elimina o modifica alguna información, ya que el ojo humano no nota estas modificaciones del color, pero si lo notase en el brillo.

Luego divide cada canal en cuadros de 8x8 pixeles y a cada cuadro por separado le aplica la transformada discreta bidimensional del coseno y cuantificación perceptual que lo que realiza es suavizar las variaciones bruscas de brillo y color, y ya por último se aplica un algoritmo de compresión sin pérdida el cual se explica líneas adelante, el algoritmo de Huffman.

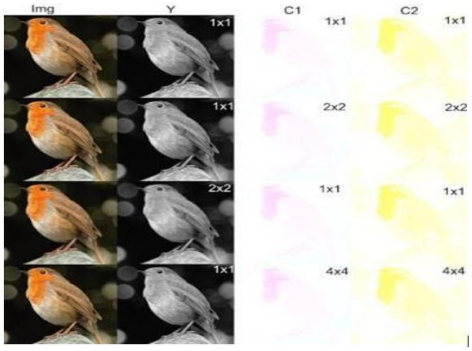

Figura 5: Funcionamiento compresión JPEG

3.3. Alternativas de compresión de imágenes sin pérdida

3.3.1. Codificación de Huffman

Es un procedimiento que permite asignar a los diferentes símbolos a comprimir, un código binario. Este algoritmo crea un árbol de nodos; primero se debe establecer un orden prioritario, el más importante es el símbolo que aparece con menor frecuencia en la cadena, luego, se eliminan los dos símbolos más prioritarios, construyendo así un nuevo "padre" que es el resultado de la suma de las frecuencias eliminadas y se ubica nuevamente en la cola de prioridad, iterativamente se realiza este proceso hasta que sólo quede un elemento, así queda construido el árbol. Luego, para asignar el código binario se contarán los pasos efectuados para llegar a cada símbolo del árbol (movimiento a la izquierda=0, a la derecha=1), obteniendo el valor de cada uno y por último reemplazandolos en la cadena.

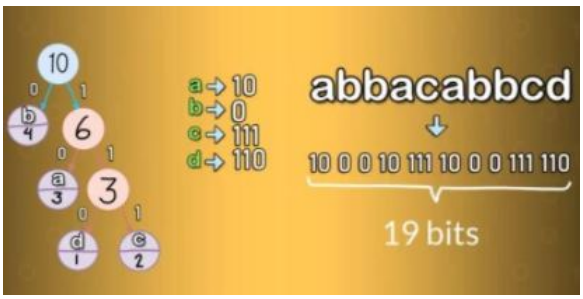

Figura 6: Ejemplo codificación de Huffman

3.3.2. Transformación de Burrows

La transformación de Burrows-Wheeler reorganiza una cadena de caracteres en series similares. Es un algoritmo que prepara los datos para su posterior uso con técnicas de compresión, al ingresar una cadena de caracteres, la transformación conmuta su orden.

El algoritmo toma la cadena de entrada y realiza todas sus rotaciones hasta que el carácter inicial vuelva a quedar en su misma posición, luego con los resultados de todas estas rotaciones los ordena por orden alfabético (los símbolos especiales como *,—,") se toman como últimos y ya una vez organizados, se toma el último carácter de cada cadena y así se obtiene la salida esperada.

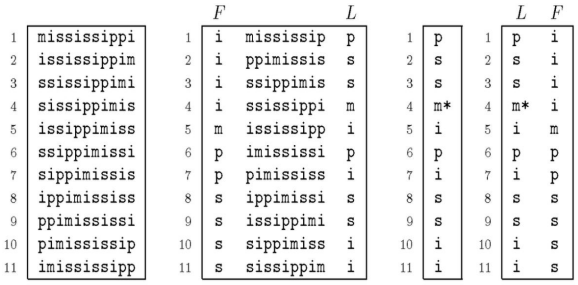

Figura 7: Ejemplo transformación de Burrows-Wheeler

3.3.3. LZ77

Es un compresor basado en algoritmo sin pérdida, este algoritmo es un tipo de codificador diccionario en el cual existen los literales, banderas y palabras claves; se empieza a recorrer la cadena y si se encuentra con un literal lo deja totalmente igual, si encuentra una bandera especifica si lo que sigue es un literal o un comprimido y si lo anterior es un comprimido (que es una especie de palabra clave) se lleva a una posición en un diccionario que arroja que bytes continúan. Por ejemplo, en una imagen las esquinas pueden ser iguales y al comprimir tales esquinas lo que hacemos es que se tenga que guardar una sola vez y por así decirlo evitar las repeticiones.

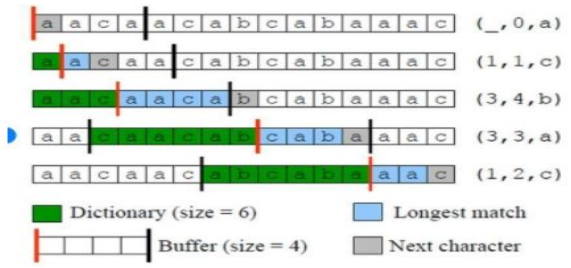

Figura 8: Ejemplo compresión con LZ77

3.3.4. LZ78

El algoritmo LZ78 usa una estructura de datos, específicamente trie, basada en diccionario para comprimir los datos. Este nació para mejorar el rendimiento respecto al LZ77, es por así decirlo, una segunda versión mejorada.

Respecto al funcionamiento para la compresión, se lee una cadena carácter por carácter y verificamos si el nodo actual (Cada nodo está marcado con el índice del diccionario) tiene algún borde de salida que contenga al carácter leído; en caso positivo configuramos el nodo actual como el nodo encontrado y repetimos el proceso, en caso negativo, se crea un borde de salida con el carácter leído se crea un nodo nuevo, el cual pasa a ser el nodo raíz e igualmente se repite el proceso.

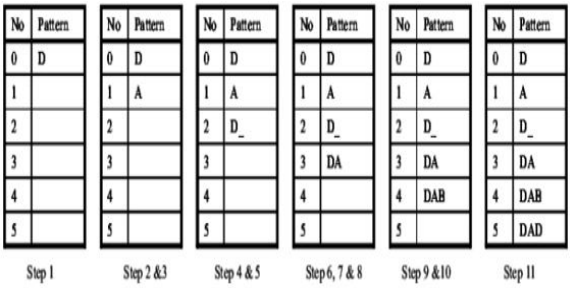

Figura 9: Ejemplo compresión usando LZ78 de la cadena "DAD DADABDAD"

4. Diseño e implementación de 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. Estructura de datos

4.1.1. Almoritmo de compresión de imágenes con pérdida

Para almacenar los datos usamos una lista enlazada (en Python deque), y dentro de ella en cada posición tenemos un diccionario que contiene el nombre del archivo y una matriz que contiene los valores de los pixeles, utilizamos el diccionario ya que nos permitía conservar los nombres originales del archivo luego de la compresión y descompresión.

Para el efecto del algoritmo de compresión con pérdida usamos sólo la matriz de píxeles.

127	94	76	57
0	128	56	87
27	115	110	95
123	112	82	53

Figura 10: Matriz de 4x4, que contiene los píxeles de cada imagen

Luego de la compresión claramente la matriz se reducirá y luego de la descompresión volverá a su tamaño original.

4.1.2. Algoritmo de compresión de imágenes sin pérdida

Para la compresión y descompresión de imágenes sin pérdida igualmente que para la compresión de imágenes con pérdida usamos una lista enlazada para almacenar los archivos del dataset y en cada posición de la lista almacenamos un diccionario que contiene el nombre del archivo (esto para poder conservar el mismo nombre del archivo luego de la compresión y descompresión) y la matriz de píxeles; para el algoritmo usamos un arreglo tanto para la compresión como para la descompresión; en el caso de la compresión el arreglo tendrá en cada posición una tupla y en la descompresión un arreglo de números, en este caso píxeles de la imagen. En la siguiente figura se ilustra mejor lo explicado anteriormente:

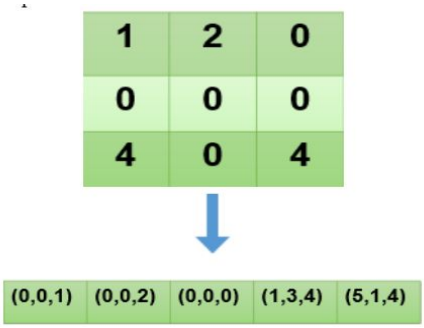

Figura 11: Matriz de 3x3 que contiene los píxeles de una imagen y arreglo de tuplas que contiene el resultado de la compresión

Como se puede observar en la imagen, la primera matriz es la original del archivo, es decir, aquella que contiene los píxeles y el arreglo bajo la flecha es el resultado de la compresión de dicha matriz, como se puede ver es un arreglo de tuplas y luego de la descompresión la matriz quedará igual a la original.

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. Algoritmos de compresión de imágenes con pérdida

Para la compresión y descompresión de las imágenes con pérdidas usamos el escalado por medio del vecino más cercano.

Este método es muy sencillo, lo único que se hace es dividir la matriz de píxeles de la imagen en submatrices de igual tamaño y escogemos cualquier píxel dentro de cada submatriz, pero siempre el de

[1] https://github.com/johnesteban/ST0245-002/tree/master/proyecto

la misma posición, es decir si en la primera submatriz escogemos el píxel de la mitad en todas las submatrices también deberemos de seleccionar el de la mitad.

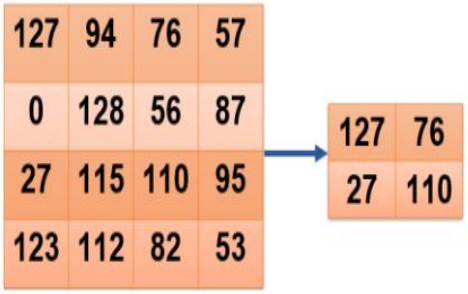

Figura 12: Compresión mediante el algoritmo del vecino más cercano

Para la descompresión lo que hacemos es duplicar el valor que obtuvimos de la compresión a lo largo del tamaño de la submatriz definido y así obtendremos nuevamente la cantidad de píxeles originales.

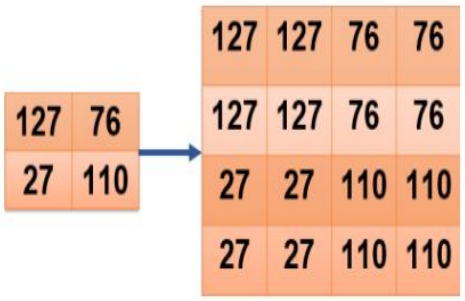

Figura 13: Descompresión mediante el algoritmo del vecino más cercano

En la siguiente imagen podemos observar los resultados de la compresión y descompresión con una de las imágenes del conjunto de datos.

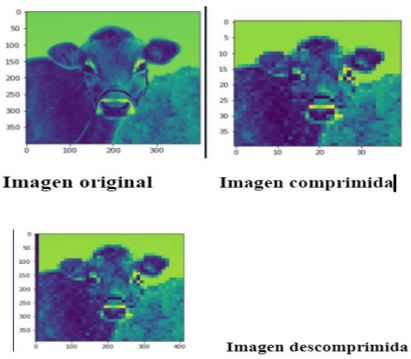

Figura 14: Ejemplo de una imagen comprimida y descomprimida

4.2.2. Algoritmo de compresión de imágenes sin pérdida

Para la compresión y descompresión de imágenes sin pérdida usamos el algoritmo LZ77.

Por un lado, para la compresión se debe convertir la matriz de píxeles en un arreglo, esto se puede hacer a través de la concatenación; luego de tener la matriz vista como un arreglo se define un tamaño de ventana y de buffer, este tiene que tener un buen balance, es decir, si se pone un buffer muy pequeño la matriz no se comprimirá nada o 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 continúa; este método consiste básicamente en encontrar repeticiones y almacenarlas en formas de tuplas que contienen toda la información necesaria (sin pérdida).

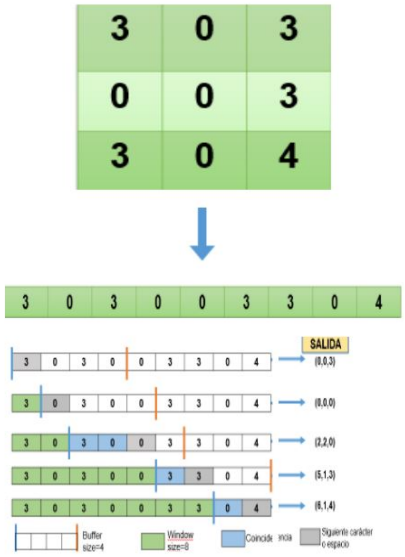

Figura 15: Compresión mediante LZ77

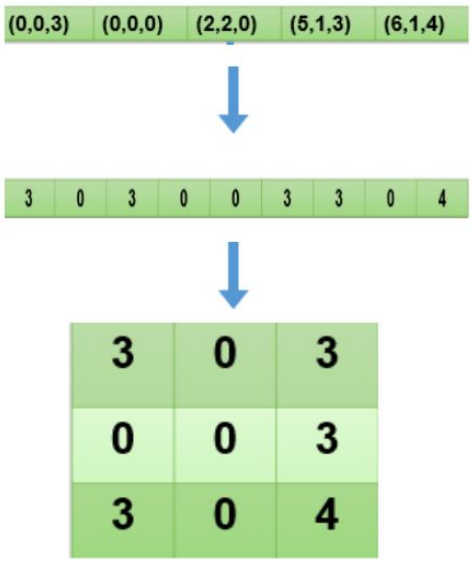

Figura 16: Descompresión mediante LZ77

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, entonces si se tiene un arreglo que contiene por ejemplo "[4,0]" y se encuentra una tupla (2,1,4) esto quiere decir que se devuelve dos posiciones atrás, en este caso al 4, toma 1 espacio que es el 4 y el espacio que sigue contiene un 4, entonces esto arrojará como resultado "[4,0,4,4]"; finalmente este arreglo se debe convertir nuevamente en matriz. En la imagen a continuación se ilustra con mayor claridad.

En la siguiente imagen se puede apreciar el resultado de la compresión y descompresión luego de aplicarle a la imagen el algoritmo del vecino más cercano y el LZ77 respectivamente:

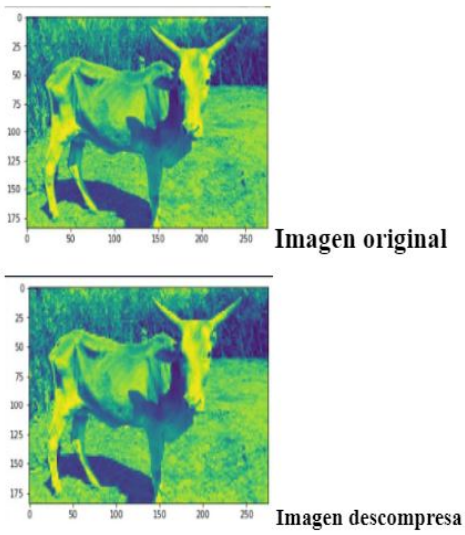

Figura 17: Resultado del algoritmo LZ77 con una imagen del dataset

4.3. Análisis de la complejidad de los algoritmos

Para lograr un buen balance entre tiempo y consumo de memoria, teniendo en cuenta que las imágenes suelen ser de tamaños muy grandes, aplicamos primeramente el algoritmo del vecino más cercano y luego el LZ77 para así trabajar con imágenes más pequeñas, por ende la imagen quedará con un poco de pérdida. A continuación explicamos la complejidad de cada algoritmo y la total.

Algoritmo vecino más cercano	**Complejidad tiempo**
Compresión	O(n*m)
Descompresión	O(n*m)

Cuadro 1: Complejidad temporal de los algoritmos de compresión y descompresión de imágenes por medio del vecino más cercano. En donde N es la cantidad de filas y M la cantidad de columnas de la matriz luego de la compresión o descompresión

Algoritmo LZ77	**Complejidad tiempo**
Compresión	O(nˆ2*mˆ2)
Descompresión	O(nˆ2*mˆ2)

Cuadro 2: Complejidad temporal de los algoritmos de compresión y descompresión de imágenes por medio del LZ77. En donde N es la cantidad de filas y M la cantidad de columnas de la matriz que contiene los píxeles de la imagen

Algoritmo final	**Complejidad tiempo**
Compresión	O(nˆ2*mˆ2)
Descompresión	O(nˆ2*mˆ2)

Cuadro 3: Complejidad temporal de los algoritmos de compresión y descompresión de imágenes por medio del vecino más cercano y LZ77. En donde N es la cantidad de filas y M la cantidad de columnas de la matriz que contiene los píxeles de la imagen

Luego de tener las complejidades temporales de todos los algoritmos, realizamos el mismo procedimiento para la complejidad en memoria.

Algoritmo vecino más cercano	**Complejidad memoria**
Compresión	O(n*m)
Descompresión	O(n*m)

Cuadro 4: Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes por medio del vecino más cercano, en donde N es la cantidad de filas y M la cantidad de columnas de la matriz luego de la compresión o descompresión

Algoritmo LZ77	**Complejidad memoria**
Compresión	O(n*m)
Descompresión	O(n*m)

Cuadro 5: Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes por medio del LZ77. En donde N es la cantidad de filas y M la cantidad de columnas de la matriz que contiene los píxeles de la imagen

Algoritmo final	**Complejidad tiempo**
Compresión	O(n*m)
Descompresión	O(n*m)

Cuadro 6: Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes por medio del vecino más cercano y LZ77. En donde N es la cantidad de filas y M la cantidad de columnas de la matriz que contiene los píxeles de la imagen

4.4. Criterios de diseño del algoritmo

Para el diseño del algoritmo nos basamos tanto en su complejidad en tiempo como en memoria, es decir, su eficiencia; como también en los fines para los cuales será usado el algoritmo, en este caso para un modelo de inteligencia artificial en el cual se transmiten las imágenes y este arroja si el ganado según la imagen está enfermo o sano.

En el caso del algoritmo de compresión de imágenes con pérdida se eligió el escalado mediante el vecino más cercano, ya que en primer lugar teniendo en cuenta el uso posterior de la imagen el re-tallado como remueve objetos de la imagen, no era viable; por otro lado, la compresión fractal

puede resultar muy costosa en algunas ocasiones, entonces por estas razones optamos por el escalado de imágenes y por último, la interpolación bilinear y el vecino más cercano poseen la misma eficiencia en tiempo y memoria, pero, mientras que la interpolación bilinear deja la imagen borrosa, el vecino más cercano la deja tipo pixel art; por lo que consideramos que el modelo puede identificar más fácilmente el patrón en una imagen tipo pixel art que conserva su forma a una imagen que en ocasiones podría ser demasiado borrosa.

Por otro lado, escogimos el algoritmo LZ77 ya que, por ejemplo comparándolo con el LZ78, este tiene unas mejoras respecto al consumo de memoria pero tarda más tiempo, y teniendo en cuenta que la prioridad era el tiempo descartamos este algoritmo; por otro lado el algoritmo de Huffman tiene una complejidad en tiempo de O(N*M*log(N*M)) tanto para compresión como para descompresión, por lo que pudimos notar que aunque la compresión se hace más rápida respecto al LZ77, la descompresión es más lenta y además este algoritmo usa mucha más memoria, aunque sus tasas de compresión suelen ser muy similares; por último el algoritmo de Burrows-Wheeler notamos que tiene una complejidad en tiempo de O(N*M*P) en donde N son las filas, M las columnas de la matriz y P la repetición más larga en toda la cadena ingresada, por lo que en ocasiones su complejidad puede ser mayor al LZ77 aunque su descompresión y complejidad en memoria O(N*M) si es igual al LZ77. Finalmente, nos pareció que el LZ77 era un buen algoritmo y sus resultados fueron bastante óptimos.

Y por último, aplicamos una combinación del algoritmo del vecino más cercano y LZ77 ya que, teniendo en cuenta que el conjunto de datos tenía imágenes muy grandes, esto iba a causar que la compresión tardará mucho más tiempo, entonces optamos por un buen balance reduciendo la imagen mediante el vecino más cercano para luego aplicarle el LZ77 y así causar que el algoritmo se ejecute en un menor tiempo, aunque tendrá un poco de pérdida pero consideramos que no afectará mucho la precisión del modelo. Por otro lado, para los modelos se suelen usar imágenes muy pequeñas, por lo que se podría realizar sólo con el LZ77. Y por último al experimentar con los diferentes tamaños de ventana y buffer en el algoritmo LZ77, notamos que a menor tamaño, el algoritmo se ejecuta más rápido pero comprimiendo a una menor tasa y viceversa si el tamaño de estos es más grande; por lo que realizamos el algoritmo con un buen tamaño de buffer y ventana para que corra a una tasa medianamente rápida y a la vez pueda encontrar repeticiones entre los píxeles de la imagen.

5. Resultados

5.1. Tasa de compresión

Presentamos los resultados de la tasa de compresión del algoritmo en la Tabla 7.

	Ganado sano	**Ganado enfermo**
Tasa compresión promedio	1:8	1:9

Cuadro 7: Promedio redondeado de la tasa de compresión de todas las imágenes de ganado sano y ganado enfermo

5.2. Tiempos de ejecución

En lo que sigue explicamos la relación entre el tiempo promedio de ejecución en segundos y el tamaño promedio en KB de las imágenes del conjunto de datos completo, en la Tabla 8.

	Tiempo promedio	***Tamaño promedio***
Compresión	30.9	284.7
Descompresión	0.46	284.7

Cuadro 8: Tiempo de ejecución de los algoritmos LZ77 y escalamiento mediante el vecino más cercano para diferentes imágenes en el conjunto de datos, tomamos 10 imágenes del conjunto de datos, tomamos su tiempo de ejecución y los promediamos; estos tiempos se tomaron con un buffer igual a 15 y una ventana igual a 30

	Tiempo promedio ventana50 buffer25	***Tiempo promedio ventana20 buffer10***
Compresión	89.9	25.9
Descompresión	0.47	0.44

Cuadro 9: Tiempo de ejecución de los algoritmos LZ77 y escalamiento mediante el vecino más cercano para diferentes imágenes con tamaños de buffer y ventana diferentes a la tabla anterior

De igual manera, teniendo en cuenta que los tiempos de ejecución varían considerablemente según el tamaño de la ventana, tomamos el promedio de los tiempos para las mismas imágenes de la tabla anterior con otros tamaños de buffer y ventana.

5.3. Consumo de memoria

Presentamos el consumo promedio de memoria en MB de los algoritmos de compresión y descompresión en KB en la Tabla 10.

	Consumo promedio memoria	***Tamaño promedio archivo***
Compresión	2.1	284.7
Descompresión	1.4	284.7

Cuadro 10: Consumo de memoria de los algoritmos LZ77 y escalamiento mediante el vecino más cercano para diferentes imágenes en el conjunto de datos

6. Discusión de resultados

Consideramos que si es apropiado el consumo de memoria y tiempo considerando los tamaños de las imágenes; aunque también pudimos notar que la descompresión se ejecuta muy rápidamente, mientras que la compresión tarda mucho más tiempo en el algoritmo LZ77 pero esto se puede corroborar observando la complejidad en tiempo según la notación O. La relación de compresión es adecuada ya que como aplicamos el vecino más cercano y se pierden algunos datos, no podríamos reducir este aún más ya que se podrían generar grandes pérdidas y por último consideramos que la compresión no debería afectar significativamente la exactitud del modelo.

6.1. Trabajos futuros

Mejorar el sistema de precisión donde la variedad de las imágenes sean concebidas de forma que no afecten el resultado que arroje la IA, la incrementación de los recursos para el aprendizaje de la IA ya que existen más factores que pueden determinar si el ganado está enfermo o sano más allá del color de los pixeles; el sacrificio de tiempo o memoria para la precisión de la compression podría dejarse la carga solo en la memoria ya que por avances de tecnología la memoria es un problema menos difícil que el tiempo para dar un resultado. La clasificación más específica de los problemas en el ganado sería un avance significativo ya que no diría solo si está enfermo sino que podría dar un diagnóstico de lo que padece la res. La posibilidad de crear un aprendizaje a partir de no solo imágenes sino de videos donde la IA pueda tener un entorno tridimensional de la res y así implementar el reconocimiento en tiempo real o casi real.

7. Reconocimientos

Este proyecto fue parcialmente apoyado por el estudiante de ingeniería de sistemas Victor Jaramillo, de la universidad Nacional de Colombia sede Medellín y por la ingeniera de sonido Paula Andrea Betancur Jiménez de la Universidad San Buenaventura. Agradecemos a la Universidad EAFIT por brindar conocimientos de alta calidad.

Referencias

edwardMorales,M.*Notas sobre compresión de datos.*2003. barnet

[1] Fedegan *Nuestra Historia.*https://www.fedegan.org.co/quienes-somos/nuestra-historia,2013. cisneros

[2] Andrew,W., Greatwood, C., Burghardt,T. *Visual Localization and Individual Identification*

of Holstein Friesian Cattle via Deep Learning.. Bristol,2007. wang

[3] Alzate,C. *Ganadería de precisión en vacuno de carne* Zaragoza,2020. ardila

[4] Debauche,O.,Mahmoudi,S.,Andriamandroso,A., Manneback,P.,Bindelle,J.,Lebeau,F. *Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors.* Alemania,2018. soto

[5] Doulgerakis,V.,Kalyvas,D.,Bocaj,E.,Giannousis, C.,Feidakis,M.,Laliotis,G.,Patrikakis,C.,Bizelis,I. *An animal welfare platform for Extensive Livestock Production Systems* adja

[6] Programmer click.*Explicación detallada del algoritmo de interpolación bilineal de imágenes* toro

[7] Moreno,R.*Algoritmos básicos para la compresión sin pérdidas.* 2010. ahs

[8] Mayordomo,E.*Los algoritmos de compresión de datos sin pérdida de información.*2019. toro

[9] Castro,J.H.*Análisis de algoritmos de compresión de imágenes para su implementación en dispositivos móviles.* 2014. toro

[10] Sandoval,M.*Algoritmo de compresión de imágenes de alta resolución sin pérdidas.* 2008.