ALGORITMO ÓPTIMO DE COMPRESIÓN DE IMÁGENES EN LA GANADERÍA DE PRECISIÓN

Brigith Lorena Giraldo
Universidad Eafit
Colombia
blgiraldov@eafit.edu.co

Luisa Fernanda Ciro
Universidad Eafit
Colombia
lfciror@eafit.edu.co

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

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

RESUMEN

El objetivo de este informe es analizar y solucionar el problema al comprimir, descomprimir y posteriormente clasificar imágenes para optimizar el consumo de energía, tiempo y memoria, sin que se afecte la precisión de la clasificación de salud del ganado, en el contexto de la ganadería de precisión. Esto es de gran importancia ya que da respuesta a la necesidad de la digitalización de los datos en la ganadería tradicional, teniendo en cuenta los recursos tecnológicos disponibles en una granja. Así pues, este problema no solo da respuesta a una necesidad en la ganadería de precisión, también es un reto importante en muchos campos del mundo actual, debido a la producción masiva de datos en diversos formatos (imagen, video, audio, etc.)

Para esto, se trabajó con el algoritmo de codificación de Huffman, el cual no presenta pérdida en la compresión y descompresión de los datos por lo que resulta conveniente para la precisión del análisis de salud de los animales en la ganadería de precisión, además de que este algoritmo presenta una tasa de compresión promedio de 2,8 : 1 y sus tiempos de ejecución promedio son de 2,1 s para la compresión y de 8,3 s para la descompresión.

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

1. INTRODUCCIÓN

Actualmente, en promedio el 33% de las proteínas consumidas en la dieta humana provienen de la ganadería, pero los efectos del cambio climático, como inundaciones o sequías podrían afectar la producción agropecuaria. Esto en consecuencia genera decadencia en la salud del ganado, es por ello que ha surgido el concepto Ganadería de precisión (GdP).

En la ganadería tradicional frecuentemente se toman decisiones basadas únicamente en la experiencia del productor. En cambio, en la ganadería de precisión al utilizar la recolección, análisis y reporte de datos; ahorra tiempo y permite tomar decisiones informadas con el monitoreo animal en tiempo real.

Por ello se requiere de una solución al problema que se genera al comprimir, analizar y descomprimir dichas imágenes del ganado sano y enfermo, sin perder precisión en la clasificación de estas, optimizando el consumo de batería, tiempo y memoria.

1.1 Problema

El problema radica en la creación de un algoritmo que permita la compresión, análisis y descompresión de imágenes de forma efectiva con el objetivo final de optimizar recursos como batería, tiempo y almacenamiento.

Debido a que la mayoría de los datos en las granjas necesitan ser digitalizados de forma óptima, sin dejar de lado la precisión del proceso clasificatorio, la solución a este es de vital importancia para la mejora de los registros del estado de salud del ganado, y por ende en la producción de unos de los principales alimentos en la dieta humana.

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.

Se hace una aproximación al problema de la compresión de imágenes por medio de un algoritmo de escalado de imagen, ya que este a pesar de ser un algoritmo con pérdida, en comparación con otras opciones la calidad de la imagen sigue siendo propicia para nuestro objetivo; a la vez que su característica reducción de las imágenes resulta bastante conveniente en la disminución de la memoria utilizada.

3. 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.

3.1 Almacenamiento de señales para el monitoreo de actividades alimentarias en ganado bovino

Actualmente los ganaderos buscan mejorar la calidad de la producción y mantener la rentabilidad, por esto se recurre al monitoreo constante del ganado, ya que de esta forma se puede cuantificar el comportamiento alimentario y obtener indicadores relevantes del bienestar animal. Para ello se tienen dos métodos:

1. La constante observación directa, la cual no es nada viable ni rentable, debido al elevado número de animales, así que no es un método muy eficaz y preciso.
2. El monitoreo por medio de dispositivos electromecánicos, el cual es más viable que el anterior, pero estos son propensos a fallas al momento de distinguir entre los efectos solapados de arranque y masticación.

De allí que, una alternativa práctica es el uso de la bioacústica para el estudio del comportamiento alimentario de los animales. Las razones que sustentan su uso son el elevado contenido de información presente en los sonidos y que los mismos pueden ser registrados sin perturbar el comportamiento natural del animal, siendo así el método más eficaz.

"Para desarrollar el software embebido en el dispositivo propuesto se utilizó Code Composer Studio, un entorno de desarrollo integrado propiedad de la empresa Texas Instruments. Dicho entorno, está basado en la plataforma Eclipse, la cual es una herramienta de código abierto para el desarrollo e implementación de sistemas embebidos mediante lenguaje C."

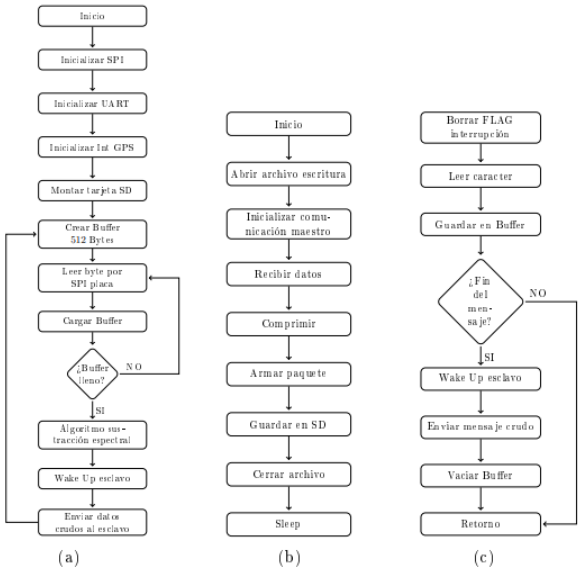

Figura 1: Diagrama de flujo

Diagrama de flujo de:

(a)La función main() del microcontrolador maestro.
(b)La función main() del microcontrolador esclavo.
(c)La rutina de servicio de interrupción del GPS.

"El preprocesamiento de las señales antes de su almacenamiento muestra resultados prometedores en pruebas preliminares. Se evaluaron distintos métodos de compresión, resultando el algoritmo CELT como el mejor para realizar dicha tarea. Además, se evaluó la utilización de distintos largos de frame de señal, dándose los mejores resultados para un frame de 256 bytes."[1]

3.2 Compresión de datos en dispositivos de cuello para ganado bovino

A pesar de las numerosas ventajas del uso de redes de sensores inalámbricos (WSN) para la recolección de datos, en la práctica en el sector ganadero no han sido comúnmente usados debido a la falta de infraestructura en los entornos de granja típicamente remotos.

Debido a esto el problema a solucionar se enfoca en obtener un sistema basado en WSN para la recolección de datos en una granja lechera, en donde como es sabido la conectividad a internet llega a ser nula o intermitente; por ello, los datos recopilados no se pueden transmitir al almacenamiento en la nube de forma oportuna, llevando a adoptar este sistema a redes tolerantes a retrasos con el fin de facilitar la transferencia confiable de los datos a la nube por medio de un dispositivo de collar, el cual consta de sensores que monitorean la salud y ubicación de las vacas, almacenando estos datos localmente en el propio dispositivo hasta que la vaca esté cerca de la puerta de enlace a la nube.

Para esto, un desafío importante es la restricción de almacenamiento de estos dispositivos de collar debido a la capacidad limitada de memoria por la gran cantidad de datos que se recopilan en un día, para abordar esta limitación se propone la compresión de datos en dichos dispositivos por medio del Edge Mining.

Algoritmo: Se utiliza Edge Mining en lugar de las técnicas tradicionales para la compresión de datos, ya que no solo reduce el volumen de los datos, sino que también sienta las bases para la retroalimentación impulsada por eventos para el prototipo.

Algorithm 1 Linear SIP for improved data collection

procedure :
Calculate new state
  Using dEWMA filtering
    $v_{x,t} \leftarrow \alpha_x * z_{x,t} + (1-\alpha_x) * (v_{x,t'} + r_{x,t'} * (t-t'))$
    $r_{x,t} \leftarrow \beta_x * (v_{x,t} - v_{x,t'})/(t-t') + (1-\beta_x) * r_{x,t'}$
Estimate new state
  Using linear extrapolation
    $v'_{x,t} \leftarrow \begin{bmatrix} 1 & (t-t') \\ 0 & 1 \end{bmatrix} v_{x,t'}$
Eventful?
  yes, if $(|v'_{x,t} - v_{x,t}| > \varepsilon_x)$
    then, store $(v_{x,t}, r_{x,t}, t)$

Figura 2: Algoritmo Edge Mining

Se usó una extensión del enfoque de Edge Mining llamada Collaborative Edge Mining en WSN, para la detección de estrés por calor en ganado lechero. Adoptando el algoritmo L-SIP sobre ClassAct y Bare Necessities para la compresión de datos, ya que permite una reconstrucción precisa de la señal en el futuro.

Los resultados obtenidos fueron favorables, llevando a la conclusión de que el problema fue abordado correctamente, ya que L-SIP proporciona una ganancia de memoria general de ~ 70% esto a su vez conduciendo a una mejora significativa en el tiempo operativo del sistema prototipo. La ganancia de memoria acumulada calculada para variables individuales fue superior al 95% durante la mayor parte del experimento con RMSE por debajo del máximo permitido en todo momento. [2]

3.3 Clasificación de datos de rastreo en animales

Actualmente la recopilación de datos en el contexto del comportamiento animal es una opción viable para la monitorización de estos. Sin embargo, existe un problema recurrente en la recopilación de estos datos, debido a los grandes conjuntos y la masiva carga de análisis e interpretación de los mismos.

Buscando alternativas adecuadas para la resolución de este problema, se introdujo el algoritmo de clasificación de K-means, para clasificar los animales. Debido a que se requería aliviar los requisitos de energía y memoria para los dispositivos de GPS, los cuales generalmente tienen limitaciones considerables.

El algoritmo K-means se utilizó para clasificar los datos de cada animal en dos categorías (activo e inactivo) sin utilizar a priori información. Se examinaron las categorías resultantes y se determinó que correspondían a períodos de actividad e inactividad claramente definidos. La clasificación se realizó sobre tres dimensiones de los datos recopilados: velocidad, ángulo de la cabeza horizontal y ángulo de la cabeza vertical.

Así que al final de la investigación se obtuvo que “K-means produjo repetidamente dos grupos a partir de los conjuntos de datos, claramente delineados en una categoría con baja velocidad y alto ángulo de cabeza, que llamamos inactivo, y uno con alta velocidad y bajo ángulo de cabeza, que llamamos activo. Esta investigación también sugiere que una tasa de muestreo de datos suficientemente alta es importante para caracterizar la utilización de la tierra por el ganado durante los períodos de actividad, mientras que la tasa de muestreo no es importante durante los períodos de inactividad. Para posteriormente ajustar la frecuencia de muestreo de datos en consecuencia. Un esquema de muestreo adaptativo de este tipo proporcionará una resolución mejorada al tiempo que conserva la memoria y la energía en los dispositivos de recopilación de datos futuros.”[3]

3.4 Uso de teléfonos inteligentes como sensores para el monitoreo del comportamiento de animales de granja

Dentro de los parámetros que los sensores permiten monitorear, el comportamiento animal es el más esencial, ya que brinda información clave sobre el estado de salud y reproductivo del animal; así pues, gracias a la disponibilidad actual de teléfonos inteligentes como iPhone, que están equipados con unidades de medición inercial (IMU) que contienen sensores de movimiento y ubicación capaces de registrar señales a alta velocidad, se propone su uso para este análisis comportamental.

Una vez recopilados los datos, Se deben identificar las variables más apropiadas y la frecuencia de muestreo adaptada para cada variable con el fin de optimizar la cantidad de datos a recolectar y almacenar en el dispositivo, eliminar las redundancias de datos en el almacenamiento local y que la compresión sea reversible sin pérdida de datos.

Se utilizaron como sensores IMU calibrados en fábrica en un iPhone 4s / 5s montado en un cabestro, se desarrolló un algoritmo de clasificación de comportamiento apropiado y una aplicación en Xamarin fue desarrollado para medir la compresibilidad de los datos reduciendo la precisión

Se elige trabajar con arquitecturas lambda ya que están diseñadas para manejar grandes cantidades de datos junto con métodos de procesamiento por lotes y por secuencias, además que esta es capaz de tratar todo tipo de datos (imágenes, video, datos temporales, datos de eventos o datos clásicos). Se usa una plataforma de alojamiento y uso compartido para tratar y explorar datos de la arquitectura IoT Lambda, la cual utiliza contenedores Apache Mesos y Docker para aislar y alojar aplicaciones.

Tanto el iPhone 4s como el 5s midieron los movimientos correctamente controlados con una precisión de 10 −3, La compresión de los datos permitió una reducción del ancho de banda consumido para su transmisión a la pasarela en un 42% en promedio. [4]

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

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

3.2.1 Tallado de costuras

El tallado de costuras desarrollado por Shai Avidan y Ariel Shamir es un algoritmo que se utiliza para el cambio del tamaño de las imágenes basado en el contenido de las mismas.

El propósito del algoritmo es poder reorientar las imágenes sin el problema de distorsionarlas en medios de varios tamaños (celulares, pantallas de proyección,etc).

Los pasos básicos del algoritmo son:

1. Se calcula el valor de energía de cada píxel (por medio de algún algoritmo).
2. En función del valor anterior, se hace la lista de costuras. Las costuras con el menor valor de energía serán las de menor importancia para el contenido de la imagen (Las costuras se pueden calcular mediante el método de programación dinámica).
3. Se eliminan todas las costuras de baja energía hasta llegar a estado ideal.
4. Se obtiene la imagen final.

El número de las líneas que se pueden eliminar de la imagen dependen de la escala a la que se desea recortar, como resultado se tendrá también que la información de la imagen como su textura de contorno de borde,etc cambiará enormemente y el gradiente también será obvio.[5][6]

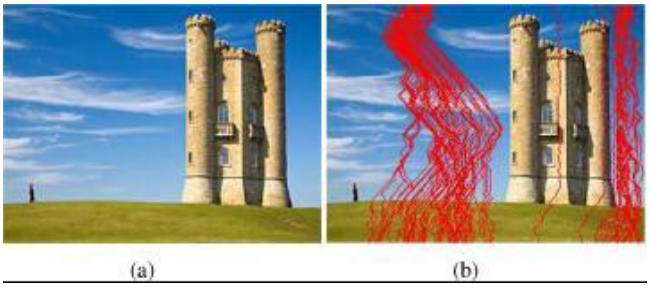

Figura 3: Tallado de costuras

3.2.2 Escalado de imágenes

El escalado de imágenes no es un algoritmo como tal, se refiere a la acción de cambiar el tamaño de una imágen digital en términos de gráficos por computadora e imagenes digitales; se pueden encontrar los siguientes casos:

1. Al escalar una imagen de gráfico vectorial se utilizan transformaciones geométricas para escalar las primitivas gráficas que componen la imagen, sin pérdida de calidad de imagen.
2. Al escalar una imagen de gráfico de mapa de bits, se debe generar una imagen con un número mayor o menor de píxeles; en el caso de disminuir el número de píxeles resulta en una pérdida de calidad visible.

Desde el punto de vista del teorema de muestreo de Nykust el escalado de imágenes se puede interpretar como una forma de remuestreo de imágenes o reconstrucción de imágenes.[7]

El escalado de imagen se puede realizar con los siguientes algoritmos:

1. interpolación del vecino más cercano
2. Algoritmos bilineales y bicúbicos
3. Remuestreo de Sinc y lanczos
4. Muestreo de caja
5. mipmap
6. Métodos de transformada de Fourier
7. Interpolación dirigida por bordes
8. hqx
9. Vectorización
10. Redes neuronales convolucionales profundas

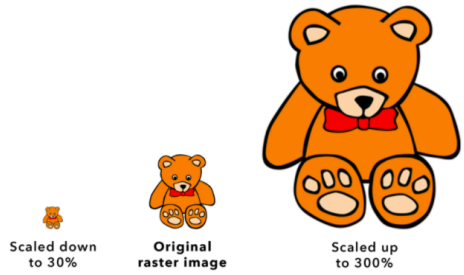

Figura 4: Escalado de imagenes

3.2.3 Transformación de coseno discreto

La DCT se aplica a codificación de imagen, desde un punto de vista de reducción de ancho de banda o compresión de datos. El objetivo es conseguir que una imagen o secuencia de imágenes, se traslade a un dominio transformado de tal forma que se reduzca el ancho de banda para la transmisión o los requerimientos para el almacenamiento; de tal forma que la subsiguiente recuperación de la imagen o secuencia de imágenes mediante la transformada inversa, no presente una distorsión perceptible.

Una imagen de tamaño NxN es dividida generalmente en bloques de tamaño LxL, por simplicidad las columnas y las filas de una imagen y de un bloque, se supone que son de un mismo tamaño.Gracias a esta división, la complejidad del hardware se reduce considerablemente en comparación con la DCT bidimensional de un cuadro completo. Después de dividir la imagen en bloques de tamaño LxL en el dominio transformado, el selector descarta algunos de los coeficientes tanto de forma adaptativa como fija; este descarte de los coeficientes de alta frecuencia en el dominio de la DCT bidimensional implica la supresión de las imágenes base correspondientes a la imagen original.[8]

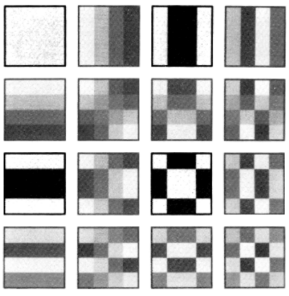

Figura 5: Transformación de coseno discreto

3.2.4 Compresión Fractal

Se basa en comprimir imágenes utilizando el hecho de que muchas imágenes “naturales” presentan autosemejanza. Este código de compresión no guarda pixeles, por lo que es libre de escalas y se puede descomprimir a cualquier escala sin tener problemas de resolución.

Dada una imagen f que queremos codificar:

1. Partimos a f en una serie de celdas rango {Ri} que deben cubrir por completo a f y no se deben superponer.
2. Cubrimos a f con un conjunto de celdas de dominio{Di}. Entre mayor sea el número de las Di's Mejor será el resultado final, pero la compresión será más tardada.
3. Para cada Ri, buscamos en el conjunto de las {Di}la celda dominio y la transformación wi que mejor la cubran. Las wi consisten en una rotación, una reducción y un ajuste de brillo y contraste. Decimos que una celda Di cubre a una Ri si la dRMS entre ambas es menor a una tolerancia dada.
4. Si el ajuste no fue lo suficientemente bueno según la tolerancia, dividimos a la celda Ri en cuatro subceldas y repetimos el punto anterior para cada subcelda.

Continuamos repitiendo estos pasos hasta que todas las celdas Ri estén cubiertas o que hayamos alcanzado el tamaño mínimo de las celdas rango que intentamos cubrir.[9]

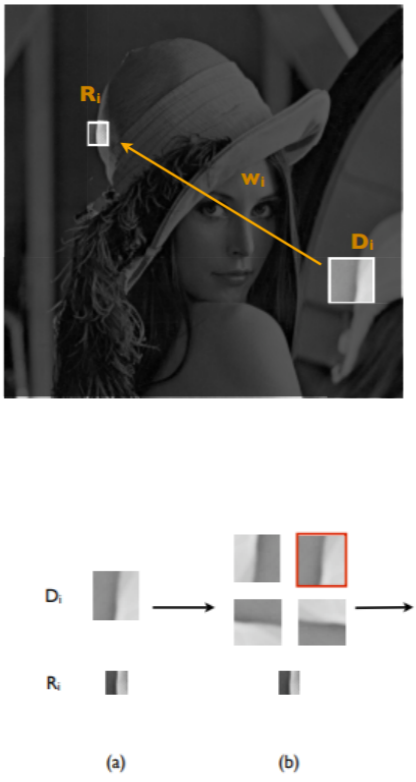

Figura 6: Compresión fractal

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 Transformada de Burrows y Wheeler

Es un algoritmo inventado Michael Burrow David Wheeler, este se basa en organizar una cadena de caracteres en series de caracteres similares, ya que suele ser fácil comprimir una cadena que tiende a tener ejecuciones de caracteres repetidos. Además de que este algoritmo permite que la transformación sea reversible sin necesidad de almacenar ningún dato adicional excepto la posición del carácter original.[10]

Para un tamaño de bloque N prefijado de antemano, el algoritmo de la transformada BWT realiza los siguientes pasos:

1. Leer la secuencia N de símbolos.
2. Construir una matriz cuadrada de lado igual al tamaño del bloque N, donde la primera fila es la secuencia original, la segunda es la secuencia desplazada un símbolo a la izquierda de forma cíclica, etc.
3. Ordenar lexicográficamente la matriz por filas. Este es el paso pesado del algoritmo y se ejecuta en un tiempo proporcional a N × log 2(N).
4. Buscar en la columna N - 1 (la de más a la derecha) la fila en la que se encuentra el primer símbolo de la secuencia original. Sea este valor i.
5. La transformada BWT de la secuencia de entrada está formada por el contenido de la columna N - 1 (la última) y el índice i.[11]

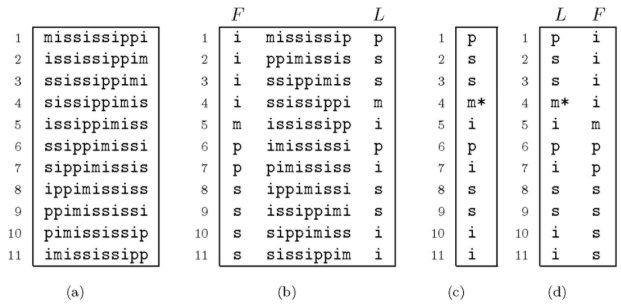

Figura 7: Transformada BWT

3.3.2 La codificación de Huffman

La idea de Huffman fue identificar los píxeles que aparecen con mayor frecuencia en una imagen y asignarles representaciones cortas. Los píxeles que ocurren con menor frecuencia en una imagen se les asigna representaciones largas.

El proceso de construcción del árbol consiste en ordenar los caracteres únicos por frecuencia relativa, de menor a mayor y de izquierda a derecha. Ponemos cada uno de estos y sus frecuencias relativas en nodos conectados por ramas. Posteriormente se forma un nodo intermedio que agrupa a los dos nodos hoja que tienen menor peso (frecuencia de aparición). El nuevo nodo intermedio tendrá como nodos hijos a estos dos nodos hoja y su campo peso será igual a la suma de los pesos de los nodos hijos. Los dos nodos hijos se eliminan de la lista de nodos, sustituyendolos por el nuevo nodo intermedio, este proceso se repite hasta que sólo quede un nodo en la lista.

El proceso de descompresión es una cuestión de traducir la secuencia de código de prefijo a valores de bytes individuales, atravesando el árbol de Huffman nodo por nodo a medida que se lee cada byte de la secuencia de entrada. [12]

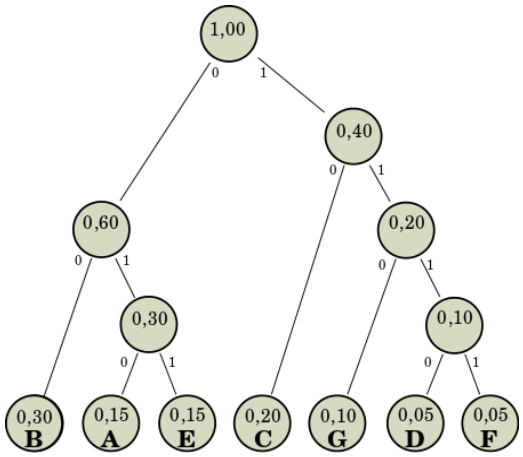

Figura 8: Codificación de Huffman

3.3.3 *LZ77*

El algoritmo LZ77 se basa en compresión de datos sin pérdida y también es conocido como lz1, siendo en teoría codificador de diccionario. LZ77 codifica y decodifica desde una ventana deslizante sobre los caracteres vistos anteriormente, y la descompresión siempre debe comenzar al principio de la entrada.

este algoritmo logra la compresión reemplazando las ocurrencias repetidas de datos con referencias a una sola copia de estos datos existentes anteriormente, en el flujo de datos sin comprimir. Una coincidencia se codifica mediante un par de números denominados par longitud-distancia.[13]

sus pasos son:

1. Se busca el matching de símbolos en el buffer look ahead en la ventana search buffer (símbolos ya procesados) y se codifica la 3-upla: (I, L, D)
2. Luego de la emisión de la 3-upla se desplaza la ventana L+1 símbolos
3. . Volver al paso 1, si no termino de leer toda la entrada

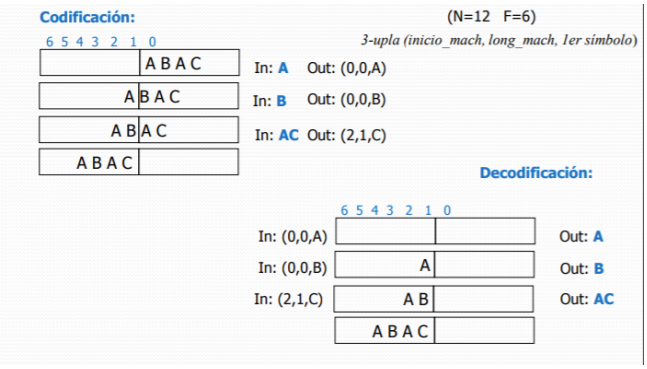

Figura 9: LZ77

3.3.4 LZ78

El algoritmo LZ78 se basa en compresión de datos sin pérdida y también es conocido como LZ2, siendo en teoría codificador de diccionario. La descompresión del lz78 podría permitir el acceso aleatorio a la entrada si se conociera todo el diccionario de antemano, sin embargo, en la práctica el diccionario se crea durante la codificación y decodificación creando una una frase cada vez que se emite un toque.

Este algoritmo logra la compresión al reemplazar las ocurrencias repetidas de datos con referencias a un diccionario que se construye en base al flujo de datos de entrada. Cada entrada del diccionario tiene el formato dictionary[...] = {index, character} , donde índice es el índice de una entrada anterior del diccionario y el carácter se agrega a la cadena representada por dictionary[index]. El algoritmo inicializa el último índice coincidente y el siguiente índice disponible. Para cada carácter del flujo de entrada, se busca en el diccionario una coincidencia. Si se encuentra, el último índice coincidente se establece en el índice de la entrada coincidente y no se genera nada. Si no se encuentra, se crea una nueva entrada de diccionario: diccionario [siguiente índice disponible] = {último índice coincidente, carácter}, y el algoritmo genera el último índice coincidente, seguido de un carácter, luego restablece el último índice coincidente e incrementa el siguiente índice disponible. Una vez que el diccionario está lleno, no se agregan más entradas. Cuando se alcanza el final del flujo de entrada, el algoritmo genera el último índice coincidente. Se debe tener en cuenta que las cadenas se almacenan en el diccionario en orden inverso, con lo que un decodificador LZ78 tendrá que lidiar. [13]

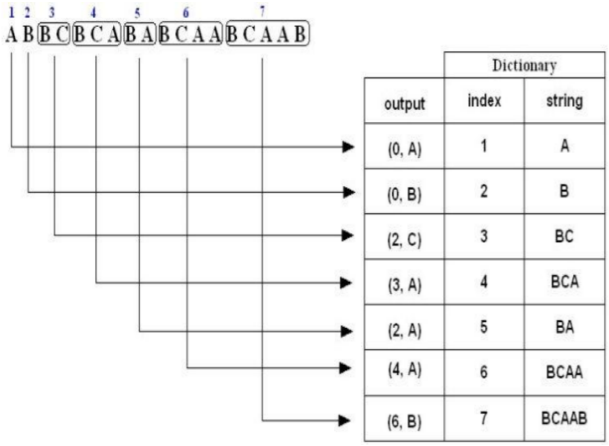

Figura 10: LZ78

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 .

4.1 Estructuras de datos

En este proyecto se usarán las tablas Hash como estructura de datos, ya que al ser un contenedor asociativo el almacenamiento y recuperación de los elementos resulta muy eficiente al tener un tiempo de recuperación constante sin que el tamaño de la tabla o el número de elementos influya en el proceso.

Una tabla hash está formada por un array de entradas, la estructura que almacene la información, y por una función de dispersión que permite asociar el elemento almacenado en una entrada con la clave de dicha entrada.

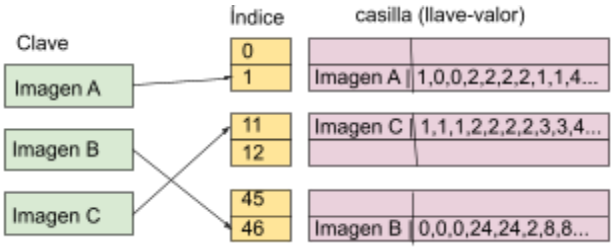

Figura 11: Tabla Hash.

Y también se usarán árboles los cuales son las estructuras de datos más utilizadas, pero también una de las más complejas, Los Árboles se caracterizan por almacenar sus

nodos en forma jerárquica y no en forma lineal como las Listas, Colas,Pilas,etc.

En este caso se usarán árboles binarios los cuales se caracterizan porque cada nodo sólo puede tener máximo 2 hijo, dicho de otra manera es un Árbol n-ario de Grado 2.

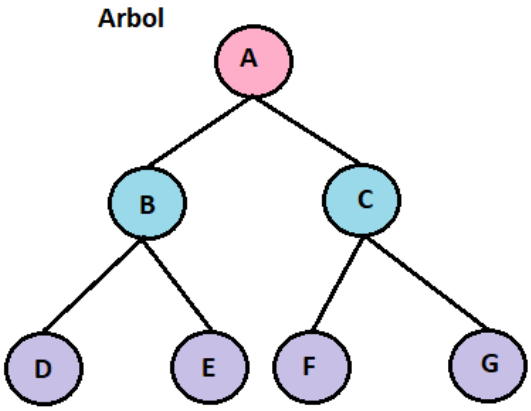

Figura 12: Árbol.

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

Para la compresión de imágenes se utiliza el escalado de imágenes como alternativa de algoritmo con pérdida, se usa el sistema de interpolación "vecino más cercano" que reduce el tamaño de las imágenes, ya que este en lugar de calcular valores promedio o tomar valores intermedios, lo que realiza es tomar el píxel más cercano y asume el valor de intensidad de este; se usó la librería de Python Opencv ya que esta nos permite leer, transformar y proyectar las imágenes a conveniencia.

10	4	22
2	18	7
9	14	25

➡

10	10	4	4	22	22
10	10	4	4	22	22
2	2	18	18	7	7
2	2	18	18	7	7
9	9	14	14	25	25
9	9	14	14	25	25

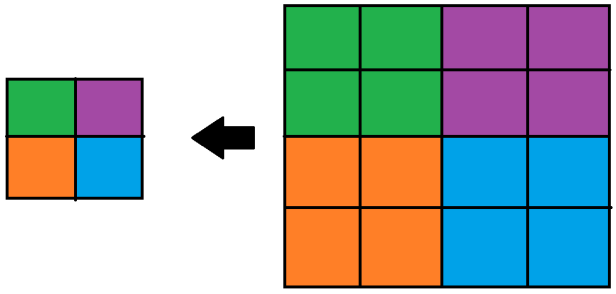

Figura 13: Escalado de la imagen mediante interpolación.

4.2.2 Algoritmo de compresión de imágenes sin pérdida

Para el algoritmo de compresión sin pérdida se implementó la codificación de huffman, para el que previamente se convirtieron los archivo csv en texto, ya que este algoritmo funciona creando un diccionario de cada carácter en la cadena de texto con un respectivo código binario, el cual se realiza obteniendo la frecuencia de cada carácter para la creación del árbol. El proceso de construcción del árbol consiste en ordenar los caracteres únicos por frecuencia relativa, de menor a mayor y de izquierda a derecha. Y así Al recorrer los nodos de este obtener los códigos binarios. Luego realiza una cadena de bits, es decir convierte la cadena de texto a una cadena de dígitos binarios para posteriormente en la descompresión poder hacer uso de esta y el diccionario que se creó anteriormente, reconstruyendo así la cadena de texto original. Ya teniendo de nuevo esta cadena, se convierte al formato csv, para proceder a pasarlo a un data frame que se almacena en una array y de allí se convierte a formato png, para así visualizar la imagen original.

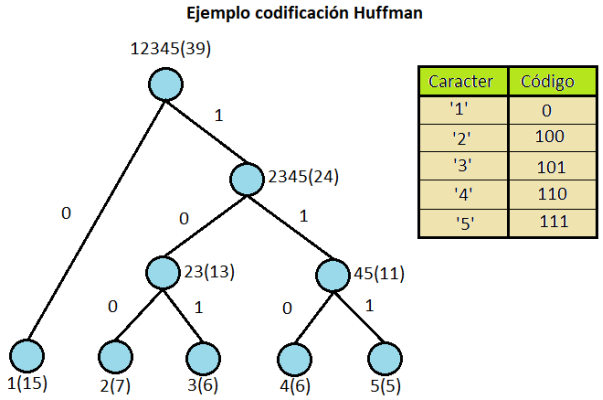

Figura 14: Codificación Huffman

4.3 Análisis de la complejidad de los algoritmos

Ya que el algoritmo pretende la compresión y descompresión de un número indefinido de archivos, para el

escalado de imágenes tanto en la compresión como en la descompresión se presenta solo 1 ciclo que depende directamente el número de archivos por lo que su complejidad es O(n). Así pues, para Huffman en la compresión tenemos ciclos que dependen de la cantidad de caracteres por lo que sería O(n), y para la descompresión tenemos 2 ciclos anidados donde el primero depende de la cantidad de digitos en un string de bits mientras que el segundo depende de la cantidad de caracteres por lo que sería O(n*m).

Algoritmo **Escalado de imágenes** **(Con Pérdida)**	**La complejidad del tiempo**
Compresión	O(N)
Descompresión	O(N)

***Tabla* 1:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. N = Cantidad de archivos csv.

Algoritmo **Escalado de imágenes** **(Con Pérdida)**	**La complejidad de la memoria**
Compresión	O(N)
Descompresión	O(N)

***Tabla* 2:** Complejidad de memoria los algoritmos de compresión y descompresión de imágenes. N = Cantidad de archivos csv.

Algoritmo **Huffman** **(Sin Pérdida)**	**Complejidad del tiempo**
Compresión	O(N)
Descompresión	O(M*N)

***Tabla* 3:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. N = cadena de caracteres. M = dígitos binarios

Algoritmo **Huffman** **(Sin Pérdida)**	**Complejidad de la memoria**
Compresión	O(N)
Descompresión	O(N)

***Tabla* 4:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. N = lista de caracteres.

4.4 Criterios de diseño del algoritmo

Al hacer la respectiva investigación acerca de los algoritmos que se implementan para la compresión de archivos, decidimos en una primera instancia usar el algoritmo de escalado de imágenes debido a que este con la función de interpolación hacía mucho más sencillo el proceso de compresión de la imagen sin perder tanta calidad y reduciendo su tamaño hasta la mitad. Ahora bien, avanzando en el proyecto terminamos realizando como opción de algoritmo sin pérdida, la codificación de Huffman debido a que este implementa un diccionario (tablas de hash) y creaba un árbol, los cuales son estructuras de datos que venimos manejando a lo largo del semestre, además de que la compresión que realiza sobre los nodos lo

hace un algoritmo bastante óptimo, también teniendo en cuenta el hecho de que es un algoritmo fácil de comprender.

Por último podemos mencionar que para ambos se realizó la conversión de formato csv a png al final de cada descompresión, para la visualización de las imágenes.

Así que ambos algoritmos resultan convenientes en la disminución de la memoria utilizada.

5. RESULTADOS

5.1 Tasa de Compresión

Presentamos los resultados de la tasa de compresión del algoritmo con pérdida en la Tabla 5.

Algoritmo con pérdida (Escalado de imágenes)	***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.

Presentamos los resultados de la tasa de compresión del algoritmo sin pérdida en la Tabla 6.

Algoritmo sin pérdida (Huffman)	***Ganado sano***	***Ganado enfermo***
Tasa de compresión promedio	2,8 : 1	2,5 : 1

Tabla 6: 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 y el tamaño promedio de las imágenes del conjunto de datos completo, en la Tabla 7 y 8.

Algoritmo con pérdida (Escalado de imágenes)	***Tiempo promedio de ejecución (s)***	***Tamaño promedio del archivo (MB)***
Compresión	0,15 s	0.82 MB
Descompresión	0.30 s	0.82 MB

Tabla 7: Tiempo de ejecución del algoritmo para diferentes imágenes en el conjunto de datos.

Algoritmo sin pérdida (Huffman)	***Tiempo promedio de ejecución (s)***	***Tamaño promedio del archivo (MB)***
Compresión	2,1 s	0.82 MB
Descompresión	8,3 s	0.82 MB

Tabla 8: Tiempo de ejecución del algoritmo para diferentes imágenes en el conjunto de datos.

5.3 Conclusión final sobre los algoritmos desarrollados

Consideramos que una ventaja en común para ambos códigos es que son muy elementales, además que tienen relación directa con varios de los temas trabajados en clase. Para el algoritmo con pérdida específicamente, la reducción en la calidad de la imágen es una clara desventaja para el objetivo final de detectar el estado de salud de los animales, por lo que el método huffman sería mucho más apropiado, sin embargo a comparación de otros métodos la desventaja de este último puede ser el tiempo, ya que este suele ser considerablemente más lento.

6. DISCUSIÓN DE LOS RESULTADOS

Si bien el consumo de tiempo puede ser mejorado con otros algoritmos, la codificación de Huffman presenta un desempeño aceptable para el contexto de la ganadería de precisión en el que se pretende emplear, en temas de memoria la tasa de compresión es buena y podría resultar muy útil en un contexto donde la conectividad no es tan rápida y no se desea manejar archivos muy pesados, en cuanto a los resultados en calidad de imagen el algoritmo no presenta pérdidas y podría facilitar el análisis de la salud del ganado.

6.1 Trabajos futuros

Para trabajos futuros podemos comentar el hecho de que no se cumplió con uno de los objetivo del proyecto en cual era la clasificación de ganado enfermo y sano mediante el uso de una red neuronal, pero aún así se pretende seguir mejorando y obtener un algoritmo más óptimo a la hora de realizar el recorrido a varios archivos csv, además de contar con varios algoritmos más (LZ77, LZW…).Y por último poder llegar a implementar la transformación de coseno discreto o la comprensión de ondeletas a futuro.

RECONOCIMIENTOS

Queremos agradecer a Kevin Sossa quien nos ayudó cuando necesitábamos de su ayuda o requerimos resolver alguna duda.

REFERENCIAS

1. Chelotti, J., Arrasin, C., Vanrell, S., Rufiner, H. and Giovanini, L. Desarrollo e implementación de un dispositivo de adquisición y almacenamiento de sonidos para ganadería de precisión. *VI Congreso Argentino de AgroInformática,* (Buenos Aires, 2014), 135-149.
2. Bhargava, K., Ivanov, S., Donnelly, W. and Kulatunga, C. Using Edge Analytics to Improve Data Collection in Precision Dairy Farming. *IEEE 41st Conference on Local Computer Networks Workshops,* (Dubai, 2016).
3. Schwager, M., Dean, A., Butler, Z., and Rus,D.Computers and Electronics in Agriculture 56 (2007) 46–59.
4. Debauche, O., Mahmoudi1, S., Andriamandroso, AHL., Manneback, P., Bindelle, J. and Lebeau, F. Cloud services integration for farm animals' behavior studies based on smartphones as activity sensors. *Journal of Ambient Intelligence and Humanized Computing*, 10, 4651–4662.
5. Wikipedia. 2020. Seam Carving. Wikipedia, the free encyclopedia. Retrieved from: https://en.wikipedia.org/wiki/Seam_carving.
6. Programador Clic. 2021. Algoritmo de talla de costura. Retrieved from: https://programmerclick.com/article/4510360417/
7. Wikipedia. 2021. Image Scaling. Wikipedia, the free encyclopedia. Retrieved from: https://en.wikipedia.org/w/index.php?title=Image_scaling
8. Transformada discreta del coseno (DCT). Retrieved from: https://signalsprocessingup.files.wordpress.com/2011/12/dct.pdf
9. Pérez, S. Compresión fractal de imágenes. Retrieved from: http://www.red-mat.unam.mx/foro/volumenes/vol029/Compresion_Fractal.pdf
10. Wikipedia. 2021. Burrows-Wheeler Transform. Wikipedia, the free encyclopedia. Retrieved from: https://en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform
11. La transformada de Burrows-Wheeler. Transformada directa. Retrieved from: http://www.hpca.ual.es/~vruiz/docencia/doctorado/html/texputse73.html
12. Lezana J. 2017. Comprensión de imágenes codificación de Huffman. Revista de Educación Matemática. Volumen 32(1), páginas 25 – 36.
13. Wikipedia. 2021. LZ77 and LZ78. Wikipedia, the free encyclopedia. Retrieved from: https://es.qaz.wiki/wiki/LZ77_and_LZ78