SISTEMA DE COMPRESIÓN DE IMÁGENES PARA ANÁLISIS DE SALUD ANIMAL

María José González Peláez
Universidad Eafit
Colombia
mjgonzalep@eafit.edu.co

Jhordan Alexis Pabón Vargas
Universidad Eafit
Colombia
japabonv@eafit.edu.co

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

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

RESUMEN

El ámbito ganadero se ha visto desde hace mucho tiempo afectado debido a las enfermedades en el ganado, estas pueden propagarse rápidamente y tener repercusiones irremediables en los animales; esto directamente afecta al sector ganadero, ya que implica una pérdida para ellos. Las enfermedades más comunes son controladas oficialmente para evitar su propagación, pero existen muchas otras que no tienen ningún tipo de control, por lo cual, se pasan por desapercibidas o se desconocen de estas.

Muchas de las enfermedades no son identificadas a tiempo y se agudiza el problema con la desinformación que hay sobre el tema. Resulta bastante importante encontrar una alternativa para la solución de este problema, para contar con la disminución de las enfermedades y sus complicaciones, y a su vez aumentando la salud del ganado y el desarrollo positivo del sector.

Para el desarrollo del proyecto utilizamos dos algoritmos: K-nearest neighbor(KNN) y LZ77. El algoritmo KNN entra en el grupo de algoritmo de compresión con pérdidas; este algoritmo funciona seleccionando un píxel entre un grupo de cuatro y así continua con toda la imagen, la tasa de compresión de este algoritmo es 2:1 ya que se eliminan la mitad de las filas y columnas que había originalmente; el tiempo de ejecución promedio de este algoritmo es aproximadamente 2.20 segundos. El algoritmo de compresión sin pérdidas LZ77 crea una ventana la cual tendrá un tamaño determinado dependiendo de cuanto se desee comprimir la imagen, ya que entre mas grande sea el tamaño de la ventana, más compresión habrá; el tiempo de ejecución promedio de este algoritmo es aproximadamente 11 segundos.

Palabras clave

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

1. INTRODUCCIÓN

La ganadería se ha visto bastante afectada desde hace mucho tiempo debido a las enfermedades que pueden contraer los animales, las repercusiones de estas pueden ser severas, traduciéndose en un potencial riesgo para los animales, salud pública y el sector ganadero.

Debido a esto se crea la necesidad de buscar una solución para la reducción de estas problemáticas. Como resultado se desarrollará un algoritmo que logre la clasificación del animal con respecto a su estado de salud; se debe tener en cuenta las condiciones de las áreas ganaderas, por lo cual, deberá ser un algoritmo que sea capaz de comprimir imágenes del ganado de una manera eficaz y que a partir de estas logre darse una catalogación precisa.

1.1. Problema

El algoritmo que se tiene planteado como solución frente a la problemática de las enfermedades del ganado, que genera un diagnóstico certero sobre la salud de los animales; sería bastante comprometedor y de igual manera resultaría de gran utilidad para la identificación de enfermedades.

No obstante, el algoritmo planteado podría resultar complicado e ineficiente en el contexto real en el que se trata, ya que ejecutarlo requeriría de diversos factores como lo es la cobertura y altos procesos de datos; por ende, es posible que en un área ganadera no sea fácil contar con ellos.

Por esto, ahora se debe buscar un algoritmo que logre comprimir imágenes a tal punto que sea posible realizar el pro- ceso desde lugares con poca cobertura, sin una alta congestión de datos y que de igual manera funcione eficaz y correctamente.

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.

Para la solución de la problemática, se utiliza el algoritmo KNN, el cual escoge entre un grupo de cuatro pixeles, solo uno. Se escoge este algoritmo ya que cuenta con un tiempo de ejecución muy bueno (0.5 segundos aproximadamente) y reduce la calidad de la imagen a la mitad, haciendo que las imágenes no sean muy pesadas y se logra la posibilidad de analizar y correr el algoritmo con una red no tan buena.

1.3 Estructura del artículo

En lo que sigue, en la Sección 2, presentamos trabajos relacionales 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 Un sistema de monitoreo de la salud animal basado en el conjunto de protocolos Zigbee
En este articulo nos exponen un sistema de monitoreo animal que recolecta una gran cantidad de información acerca del ganado como puede ser la rumia, la temperatura corporal, la frecuencia cardiaca, la temperatura y la humedad circundantes, el sistema utiliza protocolo Zigbee y un microcontrolador PIC18F4550. Con esto se buscaba solucionar el problema global que estaban teniendo los ganaderos referentes a la salud del ganado bovino por culpa del aumento constante de las temperaturas en la troposfera, causando enfermedades como la fiebre aftosa, la peste porcina, la enfermedad de las vacas locas. [1]

2.2 Un sistema integrado de control de la salud del ganado
En este articulo podemos darnos cuenta de que las técnicas clínicas se hacen insuficientes para controlar la salud del ganado ya que solo proporcionan información esporádica y requieren una gran inversión de tiempo y experiencia veterinaria. Por lo que nos presentan un sistema sofisticado el cual es capaz de evaluar continuamente la salud del ganado de forma individual, recopilar estos datos e informar de los ganaderos podría generar grandes beneficios, ya que mejoraría la salud individual del ganado y ayudaría a prevenir enfermedades generalizadas. [2]

2.3 Compresión sin pérdida de imágenes AVIRIS
En este artículo se plantea el siguiente problema con su respectiva idea para la solución, se busca llegar a la compresión sin perdidas de imágenes hiperespectrales registradas por el Espectrómetro de imágenes infrarrojas / visibles en el aire (AVIRIS). Para esto se utilizan métodos DCPM adaptativos que utilizan predicción lineal, Es imperativo que estos predictores hagan uso de las altas correlaciones espectrales entre bandas. Los residuos se codifican mediante métodos de codificación de longitud variable (VLC) y la compresión se mejora mediante el uso de ocho libros de códigos cuyo diseño depende de las características de ruido del sensor. [3]

2.4 Compresión sin pérdida de imágenes de mosaico en color
La compresión sin pérdida de imágenes de mosaico de color plantea un problema único e interesante de correlación espectral de muestras R, G, B espacialmente intercaladas, se propone una técnica de codificación de Golomb-Rice adaptativa basada en el contexto de baja complejidad para comprimir los coeficientes de la transformada de paquetes de ondas de Mallat. El rendimiento de compresión sin pérdida del método propuesto en imágenes de mosaico de color es aparentemente el mejor hasta ahora entre los códecs de imagen sin pérdida existentes.[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 .

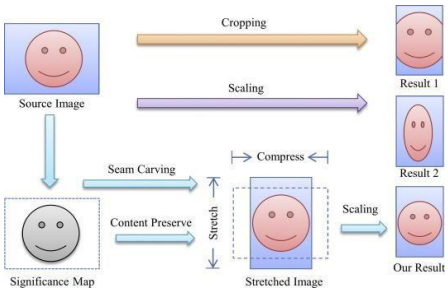
Por último, utilizando el conjunto de datos de entrenamiento, entrenamos una red neuronal convolucional para la clasificación binaria de imágenes utilizando *Teachable Machine* de Google disponible en https://teachablemachine.withgoogle.com/train/image.

3.2 Alternativas de compresión de imágenes con pérdida

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

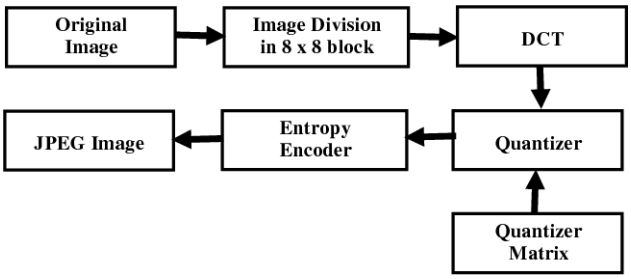
3.2.1 Tallado de costuras

El algoritmo desarrollado por Shai Avidan y Ariel Shamir, que también se le conoce como "Seam Carving", es un algoritmo que permite modificar el tamaño de las imágenes sin afectar la información del contenido ya que se encuentra consciente de este mismo; funciona estableciendo uniones que son trayectorias de menor importancia en la imagen; eliminando automáticamente uniones para reducir el tamaño de la imagen y en caso contrario, agrega uniones para extender el tamaño de la imagen. [5]



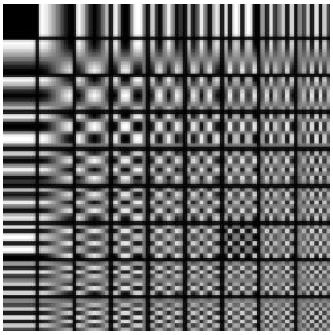
3.2.2 JPEG

El algoritmo de compresión de imágenes JPG (Join Photographic Experts Group) modifica la estructura habitual de las imágenes de píxeles de manera que cada 8 x 8 píxeles se agrupan en un bloque y se almacenan como una combinación lineal, lo que permite eliminar detalles de forma selectiva. Es utilizado en imágenes a color o en escala de grises; este algoritmo tiene la característica de originar pérdida de calidad en la imagen original usada; al aplicarlo, la imagen a color se transforma a un espacio en el que se tienen dos canales de color y uno de brillo; en los canales de color se les disminuye la resolución y se le asigna un mismo color al bloque de pixeles, procurando que sea el más parecido posible al color original. Finalmente, el algoritmo realiza la Transformada de coseno discreta (DCT) que, en pocas palabras, suaviza las variaciones bruscas de brillo y color que hay en la imagen.[6]



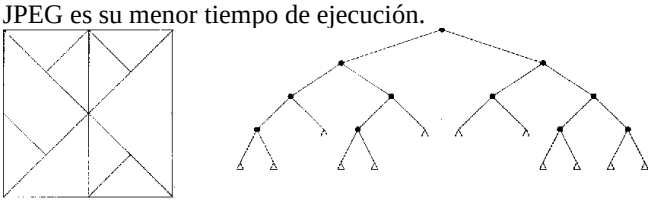
3.2.3 Transformación de coseno discreta (DCT)

La transformación de coseno discreta (DCT), es una variante de la transformada de Fourier que solo toma la parte real de los datos. Esta representa una imagen como una suma de sinusoides de diferentes magnitudes y frecuencias. La función calcula la transformación de coseno discreta bidimensional de una imagen. Este, tiene la propiedad que, para una imagen típica, la mayor parte de la información visualmente significativa sobre la imagen se concentra en sólo unos pocos coeficientes del DCT. Por esta razón, el algoritmo se utiliza a menudo en aplicaciones de compresión de imágenes.



3.2.4 B-tree triangular coding (BTTC)

Es un algoritmo de codificación que se basa en la descomposición recursiva del dominio de la imagen en triángulos rectángulos dispuestos en un árbol binario. El método es atractivo debido a su codificación rápida, O (n log n), y de- codificación, / spl Theta / (n), donde n es el número de píxeles, y porque es fácil de implementar y paralelizar. Los estudios experimentales indican que BTTC produce imágenes de calidad satisfactoria desde un punto de vista subjetivo y objetivo. Una ventaja de BTTC sobre JPEG es su menor tiempo de ejecución.



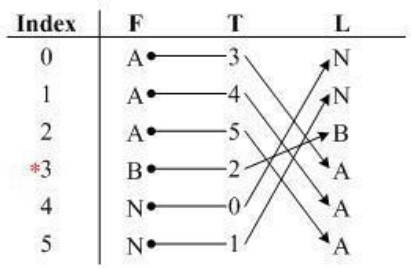
111111∗10100011∗00000000

3.3 Alternativas de compresión de imágenes sin pérdida

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

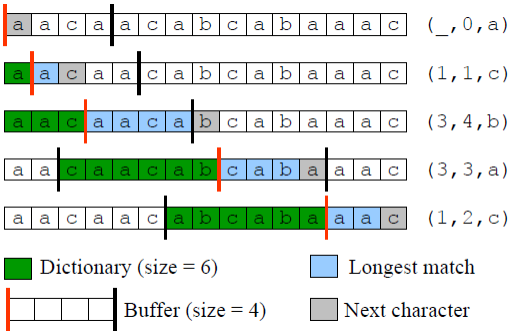
3.3.1 Transformada de Borrows Wheeler (BWT)

Es un algoritmo que toma bloques de datos, como cadenas, y los reorganiza en ejecuciones de caracteres similares. Después de la transformación, el bloque de salida contiene exactamente los mismos elementos de datos antes de que comenzara, pero difiere en el orden. La naturaleza del algoritmo tiende a colocar caracteres similares uno al lado del otro, lo que facilita la compresión del orden de datos resultante.



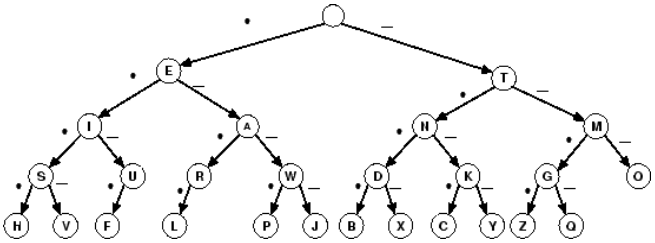
3.3.2 LZ77

El algoritmo también conocido como Lz1, es un compresor basado en algoritmo sin pérdida se utilizan cuando la información a comprimir es crítica y no se puede perder información. En este algoritmo el codificador analiza el texto como una secuencia de caracteres, mediante una ventana deslizable compuesta por dos partes; un buffer de anticipación que es la opción que está a punto de ser codificada y un buffer de búsqueda, que es la parte dónde se buscan secuencias iguales a las existentes en el buffer de anticipación. Para codificar el contenido del buffer de anticipación, se busca la secuencia igual en el buffer de búsqueda y la codificación resulta en indicar esta repetición como una tripleta [offset, longitud, carácter siguiente].



3.3.3 Algoritmo de Huffman

Es un algoritmo de compresión de imagen sin perdida que consiste en la creación de un árbol
binario en el que se etiquetan los nodos hoja con los caracteres, junto a sus frecuencias, y
de forma consecutiva se van uniendo cada pareja de nodos que menos frecuencia sumen,
pasando a crear un nuevo nodo intermedio etiquetado con dicha suma. Se procede a realizar esta acción hasta que no quedan nodos hoja por unir a ningún nodo superior, y se ha formado el árbol binario



3.3.4 LZW

LZW es un algoritmo de compresión sin pérdida es una versión mejorada del algoritmo LZ78 basado en la multiplicidad de aparición de secuencias de caracteres en la cadena que se debe codificar. Su principio consiste en sustituir patrones con un código de índice y construir progresivamente un diccionario. Además, funciona en bits y no en bytes, por lo tanto, no depende de la manera en que el procesador codifica información.

	Uncompressed Output	Dictionary	Buffer	Compressed Input
a)	0			010236
b)	01	2(0,1)	0	10236
c)	010	3(1,0)	1	0236
d)	01001	4(0,0)	0	236
e)	0100110	5(0,1,1)	2	36
f)	0100110101	6(1,0,1)	3	6

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

La estructura de datos que utilizamos para hacer la compresión de imágenes fueron los árboles binarios, más específicamente el árbol de Huffman, una estructura de datos no lineal. Este se utiliza para lograr una compresión sin perdidas. Este algoritmo toma una secuencia de n símbolos y con la frecuencia de aparición asociadas a la secuencia, se produce un código para cada frecuencia, el árbol se va formando, tomando los dos primeros símbolos con menor frecuencia, se suman sus frecuencias y esa será su raíz; este procedimiento se repetirá hacia abajo con el resto de los símbolos restantes. Para obtener el código de cada símbolo, a cada rama izquierda se le asigna el número cero y a cada rama derecha se le asigna el número uno, así cada símbolo tendrá su código único, optimo y no será ambiguo.[7]

[1] https://github.com/majogonzalezp/ST0245-001/tree/master/proyecto

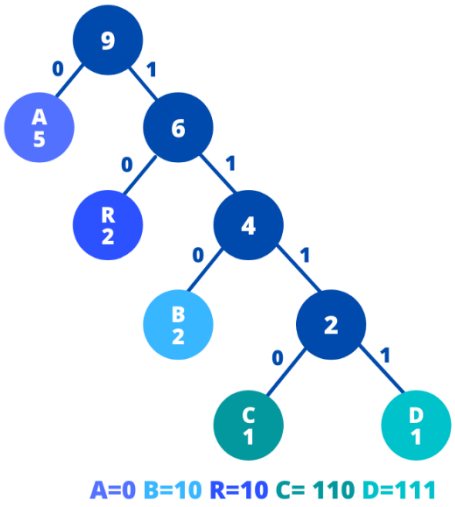


Figura 1: Árbol de Huffman generado a partir de las frecuencias exactas del texto "ABRACADABRA" el código único de cada símbolo.

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 con perdida: K-Nearest

Este algoritmo reduce el número de megapíxeles que conforman una imagen por lo que de igual manera reduce su peso, este algoritmo funciona poniendo la imagen sobre una matriz dándole a cada megapíxeles una posición de referencia, a partir de aquí se forma una matriz nueva se selecciona un valor de cada cuatro de la siguiente manera, en la primera fila se selecciona el primer valor se salta el siguiente y se selecciona el tercero, así sucesivamente hasta cubrir toda la primera fila, nos saltamos la siguiente fila que ya queda cubierta con lo seleccionado anterior mente, hacemos este proceso hasta cubrir la matriz inicial por completo y formar la matriz final

4.2.2 Algoritmo de compresión de imágenes sin pérdida

El algoritmo de compresión de imágenes sin perdida que se utilizo fue el LZ77, al cual se le ingresa una cadena de caracteres y hace una agrupación de caracteres repetidos, estos se ingresan en una tupla donde su primer elemento es el desplazamiento, que nos dirá hace cuantas posiciones de la cadena de caracteres encontró la repetición; luego está la longitud, la cual nos indicará cuantos caracteres se agrupó. Luego de la tupla, se incluye el carácter nuevo que se va a incluir.
El algoritmo se implementa en el lenguaje Python y se incluyen algunas estructuras de datos como matrices y arreglos dinámicos para la funcionalidad de este algoritmo.

4.3 Análisis de la complejidad de los algoritmos

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

***Tabla* 2:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. N y M corresponden al número de filas y columnas.

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

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. N y M son filas y columnas

4.4 Criterios de diseño del algoritmo

Este algoritmo lo diseñamos con la intención de lograr una alta eficiencia, aunque depende de varios factores, nuestro algoritmo funciona de igual manera en compresión y en descompresión. El diseño de el algoritmo fue el que decanto la elección del mismo ya que nos atraía la idea de que se basara en diccionario, es decir que encuentra coincidencias en el texto que se esta comprimiendo formando un “Historial” al que agrega los caracteres vistos anteriormente.

El y tamaño del diccionario es uno de los factores mas importantes que se deben tener en cuenta si hablamos de eficiencia de este algoritmo

5. RESULTADOS

5.1 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	23.25	0.715
Descompresión	0.08	0.461

Tabla 6: Tiempo de ejecución de los algoritmos *KNN 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	84.6	0.715
Descompresión	88.5	0.461

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.

Con un tamaño de ventana de 40 y una imagen de 0.715MB cada una tuvo la misma tasa de descompresión.

	Ganado sano	*Ganado enfermo*
Tasa de compresión promedio	1.72:1	1.72:1

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

REFERENCIAS

1. Kumar, G. Hancke. A Zigbee-Based Animal Health Monitoring System. IEEE Sensors Journal. Volume: 15. 2014.
2. K. Smith, A. Martinez, R. Craddolph , H. Erickson, D. Andresen, S. Warren. An Integrated Cattle Health Monitoring System. International Conference of the IEEE Engineering in Medicine and Biology Society. 2006
3. R. Roger; M. Cavenor. Lossless compression of AVIRIS images IEEE Transactions on Image Processing. Volume: 5. 1996.
4. N. Zhang; X. Wu. Lossless compression of color mosaic images. IEEE Transactions on Image Processing. Volume: 15. 2006.
5. Platón. Algoritmo de tallado de costuras: Una forma aparentemente imposible de cambiar una imagen. Big data. 2020
6. Programador click: Definición y construcción de árbol Huffman. Retrieved from https://programmerclick.com/article/65711865296