# **Optimización de consumo de memoria y tiempo en ganadería de precisión con compresión de imágenes**

|  |  |  |  |
| --- | --- | --- | --- |
| Miguel Ángel Chacón López  Universidad EAFIT  Colombia  machaconl@eafit.edu.co | Luisa María García Salazar  Universidad EAFIT  Colombia  lmgarcias@eafit.edu.co | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

# **RESUMEN**

En esta investigación estudiaremos diferentes algoritmos de compresión que den solución a la cuestión del ahorro de batería y recursos en la gdp para la clasificación de la salud animal, el cual es un tema bastante importante debido a que al ser la ganadería de precisión una industria tan competitiva y en que la diferencia de resultados radica en el uso de los datos, lo ideal es buscar una forma óptima de aprovechar esta información con el menor gasto de recursos como la batería y tan rápido como sea posible. Algunos de los problemas relacionados a esta situación son la tarea de encontrar los mejores procedimientos para la compresión de datos y también cómo funciona la clasificación de la salud animal en la ganadería de precisión.

## **Palabras clave**

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

# **1. INTRODUCCIÓN**

Los productos generados por el ganado son una parte fundamental de nuestra dieta debido a los importantes aportes a la nutrición humana y la es la ganadería la industria que se encarga de manejar y sacar provecho de estos productos. durante una gran cantidad de tiempo la ganadería se ha manejado de una forma tradicional, archivando los datos útiles del ganado en cuadernos o libros, pero recientemente se ha buscado un nuevo enfoque haciendo uso de herramientas tecnológicas e innovación para el manejo de estos datos, esta nueva visión es la ganadería de precisión (gdp) cuyos principales objetivos es la digitalización automática de todos los datos importantes generados y su uso para la optimización y mejora de procesos en esta industria esto mediante un refinado uso de los datos, reduciendo el consumo de energía en los equipos utilizados en las granjas y disminuyendo la dimensión de estos datos y es en este punto que los algoritmos de compresión de datos los cuales hacen posible lo anteriormente mencionado hacen presencia.

# **1.1. Problema**

Teniendo en cuenta que la ganadería es una actividad vital para la producción de alimentos, es importante que este proceso se controle y optimice, por esta razón sehace uso de la ganadería de precisión la cual permite recolectar datos de manera remota, analizar información importante, monitorear la salud animal y la utilidad que obtiene el consumidor.

La ganadería de precisión es una muy buena forma de administrar esta industria pero al trabajar con tantos datos, cámaras, sensores, etc. el ahorro de energía y tiempo toman un papel fundamental, la compresión de imágenes trae a los ganaderos un uso más óptimo de memoria y un menor tiempo de carga, ambas características muy preciadas en el contexto de la GDP ya que al estar en zonas rurales los recursos como luz, batería e internet son considerablemente limitados, al mejorar este proceso hace más fácil conseguir el objetivo por el cual se utilizan estas imágenes que es clasificar la salud de los animales utilizando tecnología e innovación, mejorando la detección y el tiempo de tratamiento de estas enfermedades y hacienda más sencillo este proceso para el ganadero.

**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 solucionar el problema hemos usado el algoritmo de compresión con pérdidas “Escalado de imágenes vecino más cercano”. Consideramos que es muy adecuado para la situación ya que es sencillo de implementar, también porque para desarrollar el código, basta con los conocimientos adquiridos hasta ahora para implementar este método. Además, reduce en muy buena cantidad el tamaño de las imágenes, cumpliendo con la necesidad de reducir recursos en un contexto donde los medios son limitados.

**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 Sistema de adquisición y análisis de información acústica para ganadería de precisión**

En esta tesis se aborda el monitoreo del comportamiento alimentario de rumiantes, lo que resulta imprescindible para conocer el estado nutricional del animal y aumentar la eficiencia en el uso de los recursos pastoriles. Para esto fue necesario diseñar, desarrollar e implementar algoritmos de bajo costo computacional que puedan operar en sistemas embebidos de capacidad de cómputo reducida. Los algoritmos desarrollados se centraron en dos tareas de diferente escala temporal pero complementarias entre sí: (i) reconocimiento de movimientos mandibulares (arranque, masticación y arranque-masticación); y (ii) reconocimiento de actividades alimentarias (rumia y pastoreo). Para el diseño algorítmico se exploraron e implementaron diferentes técnicas de aprendizaje maquinal y de procesamiento de señales. Como fuente de entrada de información para el sistema se utilizaron los sonidos producidos por el animal durante su alimentación. Se alcanzaron resultados similares e incluso superiores en desempeño a métodos del estado del arte, pero con menor costo computacional. Además, se diseñó y desarrolló un sistema embebido dedicado para ejecutar dichos algoritmos.[1]

## **2.2 Desarrollo de un algoritmo de compresión de datos optimizado para imágenes satelitales**

Las imágenes satelitales son cada vez de mayor tamaño, esto genera problemas a la hora de procesarlas y utilizar su información más relevante. Teniendo en cuenta también los desafíos del manejo de datos tanto en la plataforma satelital como en la tierra.  En esta investigación, se desarrolla e implementa un algoritmo de compresión con pérdida orientado a resolver esta problemática, utilizando la Transformada Discreta de Wavelets y la codificación Hoffman.[2]

**2.3 Clasificación de Datos Basado en Compresión**

En este artículo se presenta una vista general de diferentes trabajos realizados a lo largo del mundo que utilizan técnicas de compresión de datos como base para el desarrollo de un método de clasificación, estas técnicas se basan en la Complejidad de Kolmogorov y la utilización de esta para implementar una medida similar entre datos. El aporte principal de estos métodos es la no necesidad de un proceso de extracción de características para realizar la clasificación, lo cual hace que sea un método libre de parámetros, por lo que se puede aplicar a cualquier tipo de datos, ya sean texto, imágenes, audio, etc.[3]

**2.1 Identificación de ganado Holstein**

En este problema buscaban construir un algoritmo que identificara el ganado Holstein Friesian. Usaron herramientas (no intrusivas) como drones para obtener imágenes y videos, también canalizaciones de visión de computadora que usan redes neuronales profundas.

El algoritmo usado es la adaptación R-CNN de la red VGG CNN-M 1024, este se basa en la biblioteca de aprendizaje profundo de Caffe desarrollada por Jia et al.[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**

Este algoritmo a diferencia de la mayoría está basado en la energía para adaptarse al recorte, posteriormente, en función de este valor de píxel se emplean 8 dominios conectados para la programación Dinamica para obtener el valor mínimo, aplicando este algoritmo columna por columna o fila por fila se obtendrá una línea de energía que en realidad es la fase. la línea de píxel con el valor de pixel más pequeño de dos columnas o filas adyacentes se elimina de la imagen original.

Los pasos básicos del algoritmo son los siguientes:

1. asigna un valor de energía a cada píxel
2. Encuentra las ocho rutas conectadas del píxel con el menor valor de energía
3. Eliminar todos los píxeles en la ruta
4. Repita los pasos 1-3 anteriores hasta que el número de filas / columnas eliminadas alcance el estado ideal.

A continuación, adjuntaremos las imágenes que describen el proceso de programación dinámica que utiliza el algoritmo para calcular una costura optima.

Imagen que contiene Polígono

Descripción generada automáticamente

Imagen1. La fila superior no tiene nada encima, por lo que las energías son las mismas que las de la imagen de origen.

Gráfico, Polígono

Descripción generada automáticamente

Imagen 2. Para cada píxel en el resto de las filas, la energía es su propia energía más el mínimo de las tres energías anteriores. Repita hasta llegar al fondo.

Diagrama

Descripción generada automáticamente

Imagen 3. Para las energías más bajas que tengamos al final, trabaje de nuevo los mínimos para recuperar la costura con la mínima energía.

**3.2.2 Transformada del coseno discreta (DCT)**

La transformada de coseno discreta es una transformada basada en la [Transformada de Fourier discreta](https://es.wikipedia.org/wiki/Transformada_de_Fourier_discreta), pero utilizando únicamente números reales.

La dct es un operador lineal invertible definido en el dominio real de Rn a Rn. Se puede expresar de forma equivalente como una matriz de nxn. 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.

El DCT 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 DCT se utiliza a menudo en aplicaciones de compresión de imágenes.

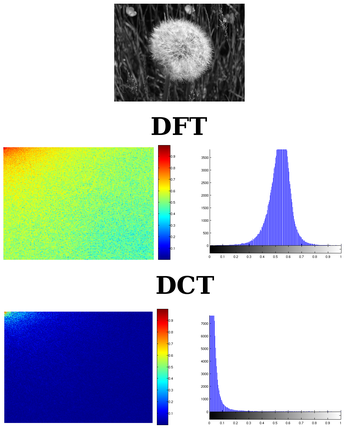


Imagen 4. En esta imagen podemos ver como la información en su mayoría se concentra en los primeros valores

**3.2.3 Compresión wavelet:**

Es un  [formato de archivo](https://es.wikipedia.org/wiki/Formato_de_archivo) [privativo](https://es.wikipedia.org/wiki/Formato_cerrado) para imágenes [raster](https://es.wikipedia.org/wiki/Raster" \o "Raster) desarrollado por la empresa [Earth Resource Mapping](https://es.wikipedia.org/w/index.php?title=Earth_Resource_Mapping&action=edit&redlink=1" \o "Earth Resource Mapping (aún no redactado)) (en la actualidad propiedad de [Leica Geosystems](https://es.wikipedia.org/wiki/Leica_Geosystems)) y que presenta unos ratios muy altos de compresión, desde 10:1 hasta de 50:1, mediante el uso de técnicas de [ondículas](https://es.wikipedia.org/wiki/Ond%C3%ADcula). Como consecuencia de esto se reduce considerablemente el tamaño de los archivos, manteniendo una alta calidad gráfica y permitiendo una rápida compresión y desccomprime transformando las imágenes al espacio wavelet usando la Transformada Discreta de Wavelet (DWT) multi-nivel. A continuación, reduce la cantidad de información de la imagen mediante cuantización, para seguidamente comprimir las imágenes del espacio wavelet. La imagen comprimida ECW se procesa línea a línea directamente a partir de la imagen original. La técnica de compresión ECW puede comprimir imágenes de cualquier tamaño usando un [algoritmo recursivo](https://es.wikipedia.org/wiki/Algoritmo_recursivo) de [segmentación](https://es.wikipedia.org/wiki/Segmentaci%C3%B3n) que no precisa del uso de almacenamiento en disco mientras se realiza la DWT.ompresión mediante un uso escaso de la memoria [RAM](https://es.wikipedia.org/wiki/Memoria_de_acceso_aleatorio).

Imagen en blanco y negro de un castillo

Descripción generada automáticamente con confianza media

Imagen 5. Esta imagen es un ejemplo de la transformada wavelet usado en JPEG 2000.

**3.2.4 Compresión fractal**

La compresión fractal es un método de [compresión con pérdida](https://es.wikipedia.org/wiki/Algoritmo_de_compresi%C3%B3n_con_p%C3%A9rdida) para [imágenes digitales](https://es.wikipedia.org/wiki/Imagen_digital), basado en [fractales](https://es.wikipedia.org/wiki/Fractal). es el más apropiado para texturas e imágenes naturales, basándose en el hecho de que partes de una imagen, a menudo, se parecen a otras partes de la misma imagen. Los [algoritmos](https://es.wikipedia.org/wiki/Algoritmos) fractales convierten estas partes en datos matemáticos llamados «códigos fractales» los cuales se usan para recrear la imagen codificada.

Forma

Descripción generada automáticamente

Imagen 6.

## **3.3 Alternativas de compresión de imágenes sin pérdida**

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

**3.3.3 Algoritmo de Hoffman**

Hoffman propuso utilizar cadenas de bits más cortas para agrupar caracteres frecuentes y cadenas de bits largas para caracteres que aparecían en menor cantidad, para esto hace uso de árboles binarios. El proceso consiste en ordenar cada carácter individual y su frecuencia en un nodo, cada nodo se ubica de izquierda a derecha y de menor a mayor.

1. Por cada carácter diferente, agruparlo en un nodo con se respectiva frecuencia.
2. Ubicar los dos nodos con menor frecuencia uno al lado del otro. Sumar ambas frecuencias para obtener el nodo padre de los dos anteriores.
3. De los nodos restantes, ubicar el de menor frecuencia al lado del nodo padre anterior.

Se repiten los pasos hasta terminar con todos los caracteres.

Por último, se sitúan ceros (0) y unos (1) en las flechas que conectan los nodos. Bajando por el árbol las flechas a la izquierda serán cero, a la derecha serán unos.

Por ejemplo, para la cadena “abracadabra” obtendremos algo como esto:

Diagrama

Descripción generada automáticamente

Imagen 7.

La cadena “abracadabra” tiene 11 caracteres, cada uno tiene un valor de 8 bits, por lo tanto, su valor es 88 bits, con el algoritmo de Hoffman ahora equivale a 23 bits porque

0 10 110 0 1110 0 1111 0 10 110 0 = 23bits

a b r a c a d a b r a

Esto representa un ahorro del 75%

El algoritmo de Hoffman es sin ambigüedades, esto quiere decir que si contamos con el diccionario y la cadena de bits podremos traducirlo a su forma original sin que se presenten confusiones entre la traducción de cada carácter.

**3.3.4 Algoritmo LZ77**

Este es un algoritmo de compresión sin perdidas que puede usarse en cualquier tipo de archivos, ya sea en imágenes, textos etc. Su funcionamiento se basa en la búsqueda de datos redundantes o repetidos para simplificar la información. Para entenderlo mejor se hace uso de una cadena de texto que queremos comprimir, sin embargo, cualquier tipo de archivo (que está representando en código binario) puede ser comprimido por este algoritmo.

Digamos que queremos comprimir la siguiente cadena de texto: “abracadabrarray”

Antes hay que tener claros algunos conceptos que usaremos

Ventana: espacio donde se encuentran una cantidad de letras, esta se amplía a medida que posteriormente encontramos caracteres repetidos.

Diccionario de tripletas: conjunto de tripletas que contienen la información comprimida del archivo, esta es útil para descomprimir más adelante la información. Para entender mejor llamaremos **X, Y, Z** a cada parte de nuestra tripleta y tienen como significado:

* X: cantidad de posiciones a retroceder para encontrarnos con el carácter en cuestión repetido.
* Y: longitud de la secuencia que hallamos repetida.
* Z: carácter siguiente a la cadena comprimida.

En este algoritmo de compresión lo que hacemos es ir identificando letras o secuencias de letras que se hallan encontrado anteriormente y ubicar la información de tripletas para que a la hora de descomprimirlo no haya errores. Todo esto se hace a través de la ventana que se amplía conforme encuentra redundancias en el texto y el carácter siguiente a esta secuencia. Siguiendo el ejemplo anterior obtenemos:

Calendario

Descripción generada automáticamente con confianza baja

Imagen 8.

**3.3.1 Transformada Burrows-Weller (BWT)**

La transformada de Burrows y Weller también conocida como compresión por ordenación de bloques es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) usado en técnicas de [compresión de datos](https://es.wikipedia.org/wiki/Compresi%C3%B3n_de_datos) como en [bzip2](https://es.wikipedia.org/wiki/Bzip2). Fue inventado por Michael Burrows y David Wheeler en 1994 y basado en una transformación inédita descubierta por Wheeler en 1983, publicada en su artículo "A Block-sorting Lossless Data Compression Algorithm".

BWT es un algoritmo que toma bloques de datos como cadenas y los reorganiza en ejecuciones de caracteres similares. Luego de la transformación, el bloque de salida contiene exactamente los mismos elementos de datos antes de que comenzara, pero difiere en el orden. El algoritmo tiende a colocar caracteres similares uno al lado del otro lo que facilita la compresión del orden de datos resultante, esta es la razón por la se utiliza en el ámbito de la compresión.

Los pasos del algoritmo cuando se tiene una cadena R son:

1. agregue un carácter especial a la cadena R
2. después de agregar el carácter especial a la cadena R, se crean todas sus rotaciones escribiéndolas en una matriz conceptual (paso a).
3. Las filas de la matriz se ordenan lexicográficamente, siendo la última columna el resultado de la transformación (paso b).

La siguiente imagen ilustra el proceso.

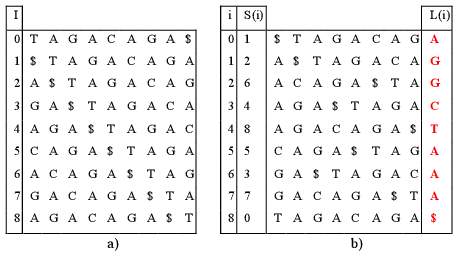


Imagen 9.

**3.3.3 LZW**

LZW es un algoritmo muy rápido tanto para la compresión como para la descompresión, 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](https://es.ccm.net/contents/57-codificacion-binaria) y no en [bytes](https://es.ccm.net/contents/57-codificacion-binaria), por lo tanto, no depende de la manera en que el procesador codifica información. Es uno de los algoritmos más populares y se utiliza particularmente en formatos TIFF y GIF. Dado que el método de compresión LZW ha sido patentado por Unisys, el que se utiliza en imágenes PNG es el algoritmo LZ77, por el que no se pagan derechos de autor.

El algoritmo elabora un diccionario, el diccionario comienza con los 256 valores de la tabla [ASCII](https://es.ccm.net/contents/55-codigo-ascii). El archivo por comprimir se divide en cadenas de bytes (por lo tanto, para las imágenes monocromáticas codificadas en 1 bit, esta compresión no es muy eficaz), cada una de estas cadenas se compara con el diccionario y se agrega si no se encuentra ahí.

En la fase de compresión el algoritmo pasa por la cadena de información y la codifica. Si una cadena nunca es más corta que la palabra más larga del diccionario, ésta se transmite.

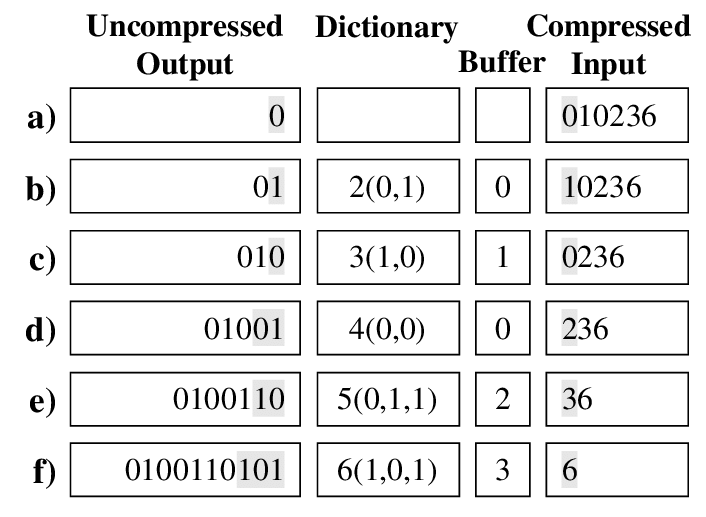


Imagen 10.

## **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]](#footnote-1).

## **4.1 Estructuras de datos**

Arreglo: estructura de datos que permite almacenar gran cantidad de información del mismo tipo. A menudo es usado para guardar varios valores en una variable y nos brinda operaciones útiles para el tratamiento de datos. En este contexto, es útil usar un arreglo porque representa perfectamente los pixeles de una imagen, además son muchos valores numéricos por almacenar y un arreglo nos permite manipular mucho mejor los datos, como por ejemplo, al querer minimizar filas y columnas.

DataFrame: estructura de datos en forma de tabla que agrupa la información del mismo tipo por columnas, es útil porque tiene integrado muchas funciones que permiten obtener información del dataFrame (como el tamaño o la cantidad de elementos). En nuestro algoritmo es implementado con el fin de simplificar la lectura de los archivos csv y tener mayor facilidad para manejarlo al recolectar datos.

Lista lee texto pasa a matriz y luego arreglo numpy, comprime, returna matriz numpy y se guarda como csv

Imagen que contiene Calendario

Descripción generada automáticamente

Imagen 11. Representación del arreglo Numpy.

**4.2 Algoritmos**

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

**4.2.1 Algoritmo de compresión de imágenes con pérdida**

El algoritmo de compresión de imágenes con pérdida “Escalado de imágenes vecino más cercano” se implementó de la siguiente forma:

El objetivo es comprimir la imagen reduciendo su tamaño y calidad. Consiste en eliminar cierta cantidad de datos tanto en filas como en columnas para que nuestra imagen sea de menor tamaño. Se piensa como en una cuadricula trazada sobre la imagen, cada cuadrado de esta es un conjunto de pixeles que se agruparan en un solo valor según el vértice más cercano a ellos, así pasaremos de varios pixeles a uno solo que conforma los más cercanos a él en la imagen original.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 12.

Para descomprimir la imagen usamos los pixeles conocidos para formar nuevos. Aquí muestra imagen aumentará de tamaño, por tanto, con respecto a la imagen comprimida habrá espacios vacíos cuyo valor es desconocido. Para formar un píxel nuevo **A,** buscamos el valor conocido más cercano a este y le asignamos el mismo.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 13.

En nuestro código leemos una carpeta que contiene los archivos csv que se guardan en un dataFrame. Convertimos el dataFrame en un arreglo Numpy y obtenemos los datos de las dimensiones, y con estos datos llamamos a la función que realiza la interpolación, reducimos el tamaño de filas y columnas para comprimir nuestra imagen reduciendo la cantidad de pixeles. La función retorna un arreglo numpy que convertimos en dataFrame. Para finalizar, convertimos el dataFrame en csv y guardamos los archivos comprimidos en una nueva carpeta.

**4.3 Análisis de la complejidad de los algoritmos**

Al analizar el algoritmo de compresión con pérdidas (interpolación del vecino más cercano) en términos de complejidad podemos notar que tiene una complejidad de O(NM). Calculamos esta complejidad a través del código, línea por lineal buscábamos su respectiva notación BigO, podemos notar en el código que en el peor de los casos nuestro ciclo anidado se recorrerá todas las veces posibles.

|  |  |
| --- | --- |
| **Algoritmo interpolación del vecino más cercano** | **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 interpolación del vecino más cercano. *(N y M representan los índices que recorren el arreglo).*

|  |  |
| --- | --- |
| **Algoritmo interpolación del vecino más cercano** | **Complejidad de la memoria** |
| Compresión | O(N) |
| Descompresión | O(N) |

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. *(N y M representan los índices que recorren el arreglo).*

|  |  |
| --- | --- |
| **Algoritmo LZ77** | **La complejidad del tiempo** |
| Compresión | O(N) |
| Descompresión | O(N\*M) |

***Tabla* 4:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes interpolación del vecino más cercano. *(N y M representan los índices que recorren el arreglo).*

|  |  |
| --- | --- |
| **Algoritmo LZ77** | **Complejidad de la memoria** |
| Compresión | O(N) |
| Descompresión | O(N) |

***Tabla* 5:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. *(N y M representan los índices que recorren el arreglo).*

**4.4 Criterios de diseño del algoritmo**

Ambos algoritmos son sencillos en cuanto a implementación y requiere un tiempo de procesamiento muy corto, lo que los hace eficientes y muy adecuados para el contexto en el que se usan,

**5. RESULTADOS**

**5.2 Tiempos de ejecución**

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

## 

|  |  |  |
| --- | --- | --- |
| Interpolación del vecino más cercano | ***Tiempo promedio de ejecución (s)*** | ***Tamaño promedio del archivo (MB)*** |
| *Compresión* | 0.486 s | 2.59 MB |
| *Descompresión* | 3.78 s | 7.6MB |

## **Tabla 6:** Interpolación del vecino más cercano

|  |  |  |
| --- | --- | --- |
| LZ77 | ***Tiempo promedio de ejecución (s)*** | ***Tamaño promedio del archivo (MB)*** |
| *Compresión* | 166.11 s | 63 MB |
| *Descompresión* | 166 s | 7.58MB |

## **Tabla 7:** LZ77

## **5.3 Consumo de memoria**

Presentamos el consumo de memoria de los algoritmos de compresión y descompresión en la Tabla 7.

|  |  |  |
| --- | --- | --- |
| Interpolación del vecino más cercano | ***Consumo promedio de memoria (MB)*** | ***Tamaño promedio del archivo (MB)*** |
| Compresión | 414MB | 2.59 MB |
| Descompresión | 7.64 MB | 7.6 MB |

## **Tabla 8:** Consumopromedio de memoria de todas las imágenes del conjunto de datos, tanto para la compresión como para la descompresión.

|  |  |  |
| --- | --- | --- |
| LZ77 | ***Consumo promedio de memoria (MB)*** | ***Tamaño promedio del archivo (MB)*** |
| Compresión | 344,5MB | 1,25 MB |
| Descompresión | 10.53MB | 7.58 MB |

## **Tabla 9:** Consumopromedio 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.

|  |  |  |
| --- | --- | --- |
|  | ***Interpolación del vecino más cercano*** | ***LZ77*** |
| Tasa de compresión promedio | 3,86:1 | 0,71:1 |

## **Tabla 10:** Promedio redondeado de la tasa de compresión de todas las imágenes de ganado sano y ganado enfermo.

## **6. DISCUSIÓN DE LOS RESULTADOS**

**6.1 Trabajos futuros**

En el futuro quisiéramos hacer mejoras en el algoritmo para obtener mayor calidad en las imágenes con menos consumo de memoria. También implementar algoritmos como la transformada de coseno que, con más conocimiento, podríamos usar para lograr mejores resultados en tiempo y memoria.

# **RECONOCIMIENTOS**

Esta investigación fue apoyada/parcialmente por Instituto Colombiano de Crédito Educativo y Estudios Técnicos en el Exterior (ICETEX) y la Beca ANDI.

# **REFERENCIAS**

|  |  |
| --- | --- |
| [1] | Chelotti, J.O. 2018. *Sistema de adquisición y análisis de información acústica para ganadería de precisión*. |
| [2] | Cruz, K.J.A. 2017. Desarrollo de un algoritmo de compresión de datos optimizado para imágenes satelitales. |
| [3] | Roman-Gonzalez, A. 2012. Clasificacion de Datos Basado en Compresion. Revista ECIPeru. 9, 1 (2012), 69–74. |
| [4] | Andrew,W., Greatwood, C., Burghardt, T. Visual Localisation and Indivi-dual Identification of Holstein Friesian Cattle via Deep Learning. IEEE Internatio-nal Conference of Computer Vision Workshop (2017), ICCVW, 2850-2859.. |

1. http://www.github.com/ ????????? /proyecto/ [↑](#footnote-ref-1)