# **ANALISIS DE DATOS Y COMPRESIÓN**

|  |  |  |  |
| --- | --- | --- | --- |
| Sebastián Diaz Gómez  Universidad Eafit  Colombia  Sdiazg1@eafit.edu.co | Jeronimo Gonzalez Gonzalez  Universidad Eafit  Colombia  Jgonzalez7@eafit.edu.co | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

# **RESUMEN**

# El problema principal que se nos presenta es el análisis de la salud de el animal por medio de un algoritmo. Pero por otro lado tenemos la compresión de estos datos, ya que la red en las zonas donde seria utilizado este algoritmo es muy inestable y limitada, por lo tanto, debemos encontrar la forma mas eficiente de realizar todos estos procesos.

Este proyecto puede muy importante para el avance tecnológico en lo que tiene que ver con la ganadería. Gracias a esto se podría automatizar este proceso y ser mucho más sencillo.

Un problema relacionado con este mismo puede ser la compresión de datos en los videojuegos, ya que se busca lograr optimizar estos de la forma mas eficiente posible, sin llegar a pedir mucha calidad en estos.

**1. INTRODUCCIÓN**

Actualmente la tecnología lleva a el hombre a buscar nuevas soluciones hacia problemas que requieren mayor esfuerzo, usando la tecnología como un método para facilitar tanto la vida de usuario comunes y corrientes como la de los ingenieros y los alrededores.

Hablando de lo anteriormente mencionado un gran problema que se maneja constantemente en el mundo de la informática son los sistemas de archivos. Las problemáticas de esto son las constante limitaciones de el sistema para almacenar una gran cantidad de los mismos.

Sin embargo, Muchos de los avances que se lograron fueron las capacidades de un sistema operativo el cual de manera eficiente logra administrar el espacio libre y acceder de manera rápida a los archivos guardados de la actualización más reciente.

# **1.1. Problema**

Como mencioné anteriormente el problema principal es lograr analizar los datos del ganado para determinar si están bien, o enfermos. Y a su vez lograr comprimir estos datos para hacer mas eficiente el proceso debido a la mala calidad de red en las zonas ganaderas.

Esto puede aportar mucho a la ganadería en cuanto a eficiencia, ya que ayuda a que estos procesos se hagan mucho más rápidos y sencillos, por lo que es un avance muy importante.

**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.

**1.3 Estructura del artículo**

En lo que sigue, en la Sección 2, presentamos trabajos relacional con el problema. Más adelante, en la Ssección 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.

**3.1 Una revisión sistemática de la biliografía sobre el uso del machine learning en la ganadería de precisión.**

El artículo que se presenta hace una revisión sistemática de la bibliográfica sobre los trabajos recientes realizados utilizando Machine Learning para el sector de la ganadería más enfocado a la ganadería de precisión. Este artículo se centra más que todo en dos áreas: El pastoreo y la salud del animal. En este artículo presenta las oportunidades para en Machine Learning en el sector ganadero, muestra qué avances ha habido en el análisis de datos, detalla el incremento de apertura de fuentes de datos.

Los algoritmos que presentan en este artículo son: Simple Logistic (SL); Logistic model trees (LMT), MLP, Naive Bayes (NB), DT, SVM, Naive Bayes tree (NBTree), Logistic Model Trees (LMT) and Squential Minimal Optimization (SMO).

El resultado de este trabajo arrojó que el Machine Learning apenas está en etapa de desarrollo y tiene diferentes desafíos tales como desarrollar modelos de diagnóstico como prevención y control de enfermedades en los bovinos en el PLF; dar autonomía a la PLF mediante ciclos autónomos de tareas de análisis de datos y meta-aprendizaje, y reunir las variables del suelo y del pasto ya que son importantes para la salud animal y de pastoreo.

**3.2** **Una plataforma de bienestar para sistemas extensivos de producción ganadera.**

El estudio actual presenta el progreso continuo del desarrollo de un sistema automatizado con un solo tipo de sensor inalámbrico capaz de registrar indicadores del bienestar del animal (es decir, información de movimiento, velocidad y geolocalización del animal) con bajo costo de implementación, basado en Deep Neural Algoritmos de reconocimiento de patrones de red. La solución también proporciona a los usuarios finales (agricultores) visualizaciones de información útiles y efectivas, para que tomen las acciones adecuadas.

De acuerdo con la información anterior, los algoritmos que presentan en este artículo son algoritmos de reconocimiento de patrones de redes neurales profundas.

El presente estudio presenta una solución para el seguimiento y monitoreo de la actividad y el comportamiento de los animales en las fincas ganaderas, obteniendo indicadores que sustentan el bienestar animal. La solución, I. un solo tipo de sensor inalámbrico (dispositivo de collar) para registrar la actividad de los animales como: el movimiento, velocidad, geolocalización, con un bajo costo de implementación, II. dispositivos de computación con capacidades computacionales, capaces de realizar procesamiento de datos desconectados y en tiempo real para el reconocimiento de patrones a través de algoritmos de redes neuronales profundas, III. Computación en la nube para datos y almacenamiento de modelos de aprendizaje profundo y IV visualizaciones utilizables y efectivas en dispositivos móviles que brindan a los agricultores información valiosa.

**3.3 Integración de servicios en la nube para el comportamiento de animales de granja basados en teléfonos inteligentes como sensores de actividad.**

Los teléfonos inteligentes, particularmente el iPhone, pueden ser instrumentos relevantes para los investigadores en comportamiento animal porque están fácilmente disponibles en el planeta, contienen muchos sensores y no requieren desarrollo de hardware. Están equipados con Unidades de Medida Inercial (IMU) de alto rendimiento y sistemas de posicionamiento absoluto que analizan los movimientos de los usuarios, pero pueden ser fácilmente desviados para analizar igualmente los comportamientos de animales domésticos como el ganado.

El estudio del comportamiento animal utilizando teléfonos inteligentes requiere el almacenamiento de muchas variables de alta frecuencia de muchos individuos y su procesamiento a través de varias combinaciones de variables relevantes para el modelado y la toma de decisiones. Transferir, almacenar, tratar y compartir tal cantidad de datos es un gran desafío. En este artículo, se propone una arquitectura de nube lambda acoplada de manera innovadora a una plataforma de intercambio científico utilizada para archivar y procesar datos de alta frecuencia para integrar los desarrollos futuros de Internet de las cosas aplicados al monitoreo de animales domésticos. Se ejemplifica una aplicación al estudio del comportamiento del ganado en pastos basada en los datos registrados con la IMU de iPhone 4s. También se logra una comparación de rendimiento entre el iPhone 4s y el iPhone 5s. El paquete también viene con una interfaz web para codificar el comportamiento real observado en los videos y sincronizar las observaciones con las señales del sensor. Finalmente, el uso de Edge Computing en el iPhone redujo en un 43,5% en promedio el tamaño de los datos sin procesar al eliminar las redundancias. La limitación del número de dígitos en una variable individual puede reducir la redundancia de datos hasta en un 98,5%.

Para el caso de este artículo, se presenta una aplicación llamada Xamarin, la cual permite medir la compresibilidad de los archivos. Sin embargo, en este artículo no se menciona algún algoritmo de compresibilidad de datos, simplemente se menciona que es debido considerar optimizarlos para mejorar el consume de batería de los dispositivos con los que se trabajó.

**3.4 Localización visual e identificación individual del ganado holstein friesian mediante aprendizaje profundo (deep learning).**

Este artículo nos presenta una demostración de que la revisión por computadora de los oleoductos se puede utilizar con arquitectura neuronal profunda sobre la detección automatizada del ganado Holstein Friesian así como la identificación individual de configuraciones agrícolas relevantes.

Demostraron que las redes estándar pueden realizar la identificación completa de los individuos en imágenes fijas de arriba hacia abajo adquiridas por medio de una cámara fija. Luego, presentamos una canalización de procesamiento de video compuesta por componentes estándar para procesar de manera eficiente el dinamismo del ganado filmado por un dron. Informamos sobre estas configuraciones, así como el contexto, la capacitación y la evaluación de todos sus componentes.

Demostraron que la detección y localización del ganado Friesian se puede realizar de forma robusta con una precisión del 99,3% en estos datos. Se evaluó la identificación individual aprovechando la singularidad del pelaje por medio equipos RGB tomados después del ordeño con 89 individuos y una precisión de 86,1%. También se evaluó la identificación a través de canalización de procesamiento de video en 46.430 cuadros provenientes de 34 videos, cada uno de aproximadamente 20 segundos de duración tomadas por el dron durante el pastoreo, con 23 individuos y una precisión de 98.1%.

## El algoritmo implementado en el presente artículo es un algoritmo de comprensión de imágenes y canalización se componentes. El diseño del algoritmo es una línea que utiliza entradas de imágenes fijas y comprende un R-CNN VGG-M 1024 (Receptor de imagen y video) de extremo a extremo para la identificación individual del ganado. También posee una antena de seguimiento y una unidad de trayectoria (KCF) única a un componente de identificación individual que utiliza redes de información complementaria

## **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.

## 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.

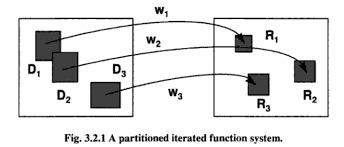
## **3.2 Alternativas de compresión de imágenes con pérdida**

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

**3.2.1 Compresión fractal**

La compresión fractal es un método de compresión con pérdida para imágenes digitales, basado en fractales. El método 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 fractales convierten estas partes en datos matemáticos llamados «códigos fractales» los cuales se usan para recrear la imagen codificada.

Con la compresión fractal, la codificación es extremadamente cara a nivel computacional debido a la búsqueda de similitudes propias. Sin embargo, la decodificación es bastante rápida. Mientras que esta asimetría lo hace poco práctico para aplicaciones en tiempo real, cuando el vídeo es guardado para distribución desde un disco, la compresión fractal llega a ser más competitiva.



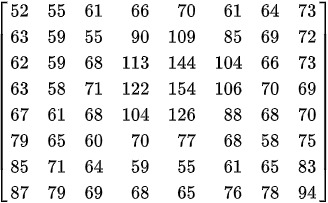
**3.2.2 Joint Photographic Experts Group (JEPG)**

JPEG utiliza una forma de compresión con pérdida basada en la transformada de coseno discreta (DCT). Esta operación matemática convierte cada cuadro / campo de la fuente de video del dominio espacial (2D) al dominio de frecuencia (también conocido como dominio de transformación).

Un modelo perceptivo basado libremente en el sistema psicovisual humano descarta información de alta frecuencia, es decir, transiciones bruscas en intensidad y tono de color. En el dominio de la transformación, el proceso de reducción de información se denomina cuantificación. En términos más simples, la cuantificación es un método para reducir de manera óptima una escala de números grandes (con diferentes ocurrencias de cada número) en una más pequeña.

El método de compresión suele tener pérdidas, lo que significa que parte de la información de la imagen original se pierde y no se puede restaurar, lo que posiblemente afecte a la calidad de la imagen.

A continuación, cada bloque de 8 × 8 de cada componente (Y, Cb, Cr) se convierte en una representación de dominio de frecuencia, utilizando una transformada de coseno discreta (DCT) de tipo II bidimensional normalizada:



**3.2.3 codificación por transformación (NTSC, PAL, SECAM)**

Codificación por transformación: Es un tipo de compresión para datos naturales como: Audio, video e imagen, la transformación conlleva perdida de información, resultando una copia de menor calidad a la que entra originalmente.

Un ejemplo de esto es la televisión, que utiliza sistemas de compresión por transformación como:

NTSC: Este es uno de los sistemas de codificación por transformación con mayor éxito – se enfocó en el formato de televisión a color. Este conocimiento permitió desarrollar in sistema que descartara mayor parte de la señal entrante, el resultado fue una señal con menos contenido que encajo en los 6MHz de señal a blanco y negro con una diferencia modulada.

PAL y SECAM: Estos sistemas utilizan métodos muy parecidos para transmitir color.

Ejemplo de codificación por compresión:

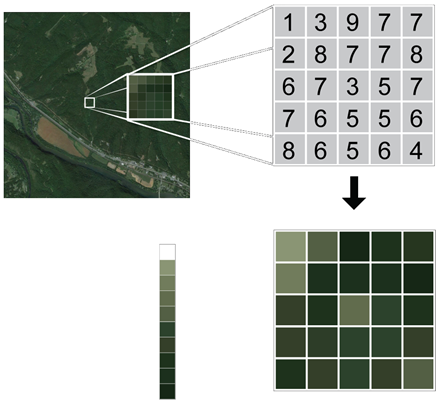


**3.2.4 MrSid**

Es un algoritmo de compresión de imágenes con perdida y un estándar abierto de compresión de imágenes raster (Estructuras de pixeles), permite mostrar archivos digitales de gran tamaño con un tiempo de carga mínimo gracias a tecnologías ondículas (Transforma las señales en gráficas y algoritmos matemáticos).

La característica predominante del formato es el teselado (Patrón de figureas) que logra una alta compresión de imágenes digitales con la pérdida mínima de detalle. El teselado da la capacidad de descomprimirse solamente aquella porción de imagen solicitada por el usuario, extrayendo y entregando únicamente los bitplanes y datos raster necesarios para construir la vista requerida.

Datos raster: También conocido como mapa de bits o plano de bits, son datos representados por medio de pixeles o puntos de color, que se pueden visualizar por medio de monitores u otros dispositivos (Tiene cierta relación con el PNG).



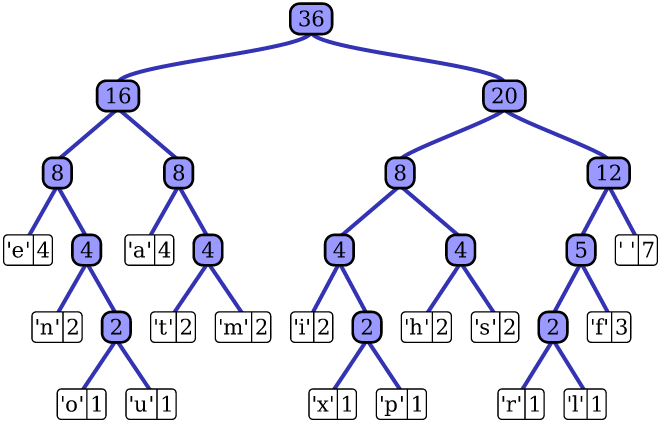
## **3.3 Alternativas de compresión de imágenes sin pérdida**

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

**3.3.1 codificación de Huffman**

Es un algoritmo usado para la compresión de datos. Se refiere al uso de una tabla de códigos de longitud variable para codificar determinado símbolo, donde la tabla ha sido rellenada de tal forma que se basa en la probabilidad estimada de aparición de cada de cada posible valor de dicho símbolo.

La codificación de Huffman usa un método especifico para elegir la representación de cada símbolo, que representa los caracteres mas comunes usando las cadenas de bits mas cortas, y viceversa.

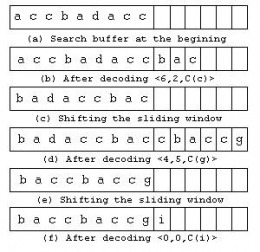


Un ejemplo de un árbol dibujado es el siguiente donde los nodos hijos finales son a los que les corresponde un símbolo y una frecuencia, y los nodos padres de estos últimos se conectan como pequeños árboles a otros nodos. Los nodos con puros números indican la suma de las frecuencias de los nodos hijos correspondientes.

**3.3.2 LZ77**

Los algoritmos LZ77 logran la compresión reemplazando las apariciones repetidas de datos con referencias a una sola copia de esos datos existente anteriormente en el flujo de datos sin comprimir.

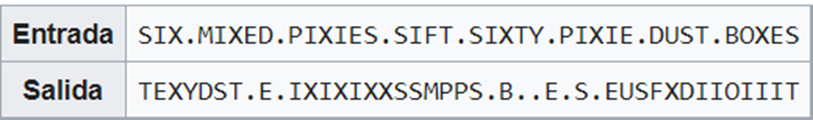
Para detectar coincidencias, el codificador debe realizar un seguimiento de cierta cantidad de los datos más recientes, como los últimos 2 kB, 4 kB o 32 kB. La estructura en la que se guardan estos datos se denomina ventana deslizante, por lo que LZ77 a veces se denomina compresión de ventana deslizante. El codificador necesita mantener estos datos para buscar coincidencias, y el decodificador necesita mantener estos datos para interpretar las coincidencias a las que se refiere el codificador. Cuanto más grande sea la ventana deslizante, más atrás podrá buscar el codificador para crear referencias.



**3.3.3 Burros Wheller**

Es un algoritmo sin perdidas, es utilizado para la compresión de datos como en bzip.2. Cuando se transforma una cadena de caracteres mediante la BTW, ningún carácter cambia de valor. Si la cadena contiene muchas sub cadenas, entonces la cadena transformada contendrá múltiples opciones en las que el mismo carácter este repetido varias veces.

Es muy útil para la compresión de archivos, ya que tiende a ser fácil comprimir una cadena que tiene una secuencia de caracteres repetidos con técnicas como: move-to-front-transform y run-length encording.



La salida es más fácil de comprimir ya que tiene muchos caracteres repetidos. De hecho, en la cadena transformada.

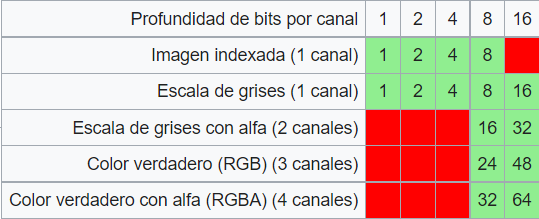
aparece un total de seis secuencias de caracteres idénticos:

XX, SS, PP, ..., II, y III, que juntos representan 13 de los 44 caracteres.

**3.3.4 PNG**

Es un algoritmo de comprensión sin perdidas para bitmaps. Este formato fue desarrollado para solventar las deficiencias del formato gif y permita almacenas imágenes con mayo profundidad de contraste y otros datos importantes.

El método de compresión utilizado por el PNG es conocido como deflación (Algoritmo sin perdidas). También tiene métodos de filtrado de información en pixeles, el mas utilizado es prediciendo el valor aproximado de los pixeles lo cual mejora la comprensión para cada línea de la imagen a comprimir. En otras palabras, este método de filtrado predice el color de cada pixel basándose en los colores de los pixeles previos a comprimir, restándolos con los actuales, Lo cual reduce el tamaño del archivo de imagen.



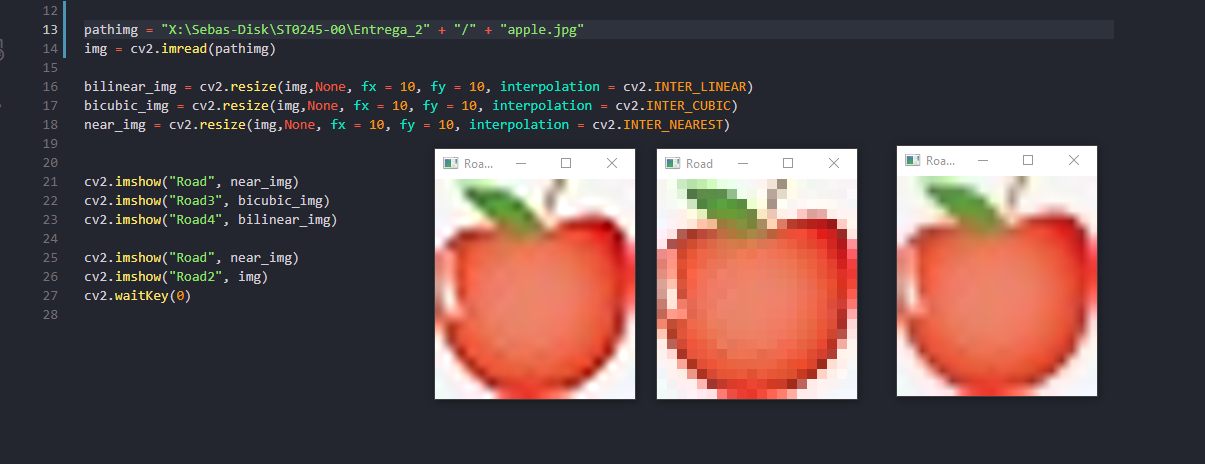
## **4. DISEÑO E IMPLEMENTACIÓN DE LOS ALGORITMOS**

## 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).

Nuestro algoritmo se compresión con perdida realizado con Python parte de el uso de la librería OPENCV la cuales cuenta con la posibilidad de comprimir imágenes de manera rápida y sencilla, usando el proceso de interpolación en 3 de los casos de selección, como son, Vecino mas cercano, Interlinear, Inter Cubico.

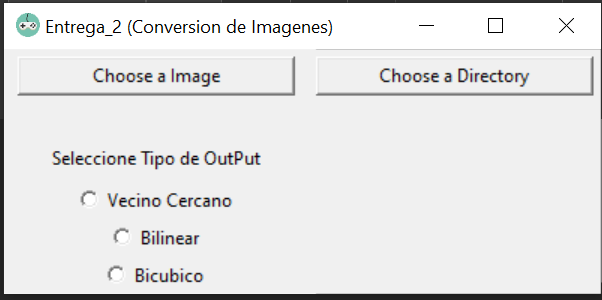
Estos usan como parámetros la asignación de la imagen a comprimir, las dimensiones que se les desean asignar para después de la compresión y que tipo de compresión se desea realizar a la imagen seleccionada.

La calidad puede variar dependiendo de las dimensiones en las que se desea la compresión, como base nosotros decidimos partir de la dimensión base de la imagen e interpolar a una perdida menor a la entregada. Pero para experimentos realizados usamos una dimensión 600x600 en la cual se observo de manera sorprendente la cantidad de cada imagen partiendo de su dimensión principal.

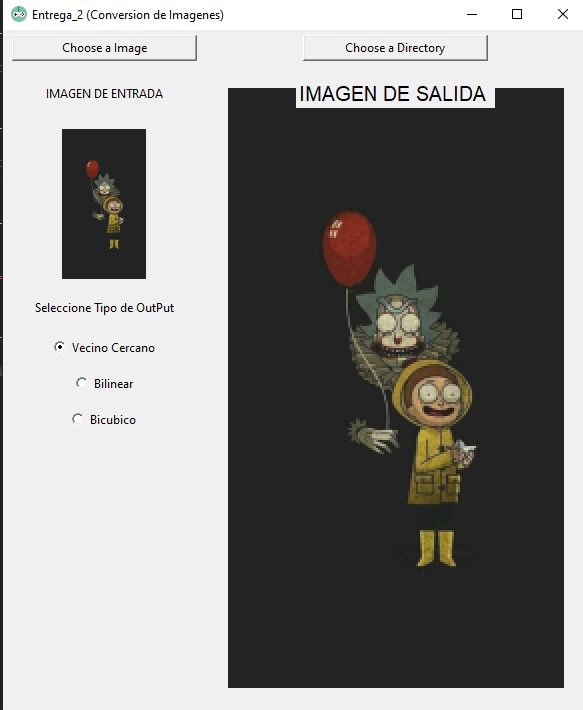


La imagen que se uso de prueba fue una manzana con dimensión 20x20 la cual su tamaño se escalo 10 veces su tamaño de forma interpolada para entrega de manera mas visible su cambio de calidad con su respectiva perdida.

El código completo se implementó una interfaz gráfica para mejorar la experiencia del usuario (No era necesario), sin embargo, decidimos implementar la posibilidad de comprimir una cantidad de imágenes posibles encontradas en un directorio, estas compresiones se almacenan en una carpeta individual para cada tipo de compresión asignada, CUBICO, VECINO CERCANO, BILINEAR.

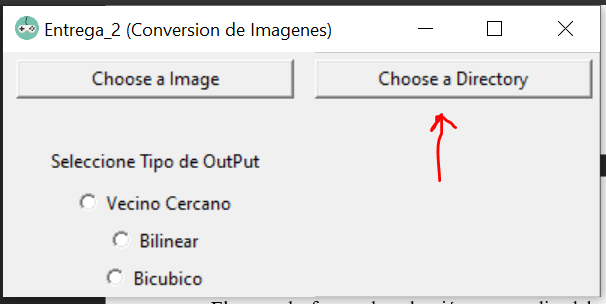


Existen 2 posibilidades de compresiones en nuestro programa, la primera es por imagen de manera individual. Logrando escoger a gusto el usuario la posibilidad de compresión de cada uno.

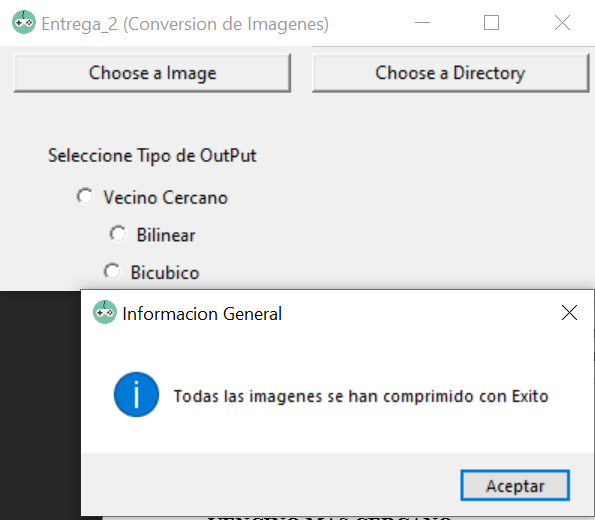


Se permite a selección del usuario el tipo de compresión que desea mostrando directamente desde la interfaz.

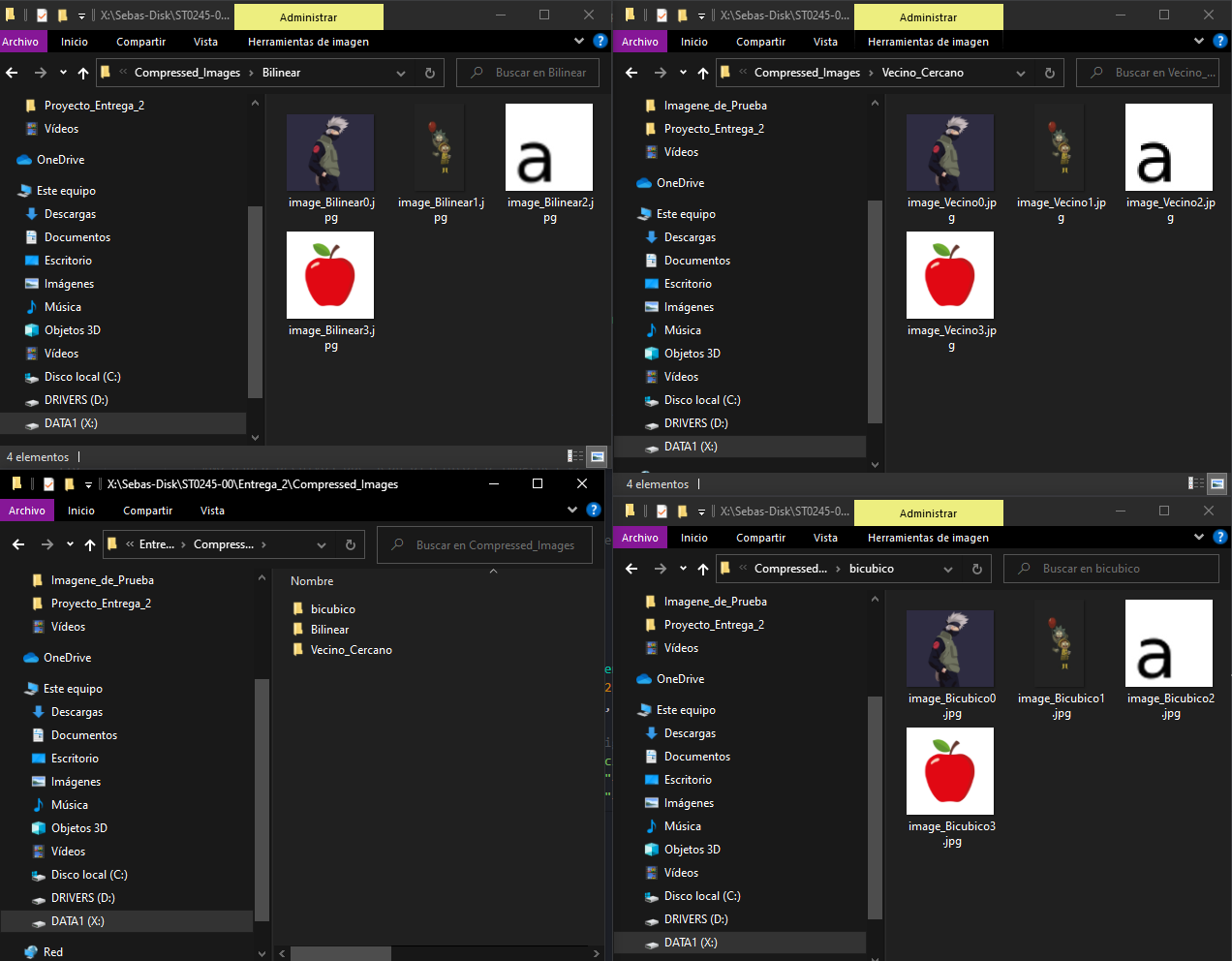
La segunda forma de selección por medio del usuario es comprimir todas las imágenes de una carpeta en sus 3 tipos de compresión con perdida.



Al momento de el usuario escoger su carpeta con las imágenes que desea comprimir, el algoritmo genera 1 carpeta con 3 subcarpetas con cada compresión de cada imagen que se encontraba en su directorio.



(DEMOSTRACION DEL PROCESO DEL ALGORITMO)

****

**4.1 Estructuras de datos**

**VENCINO MAS CERCANO**

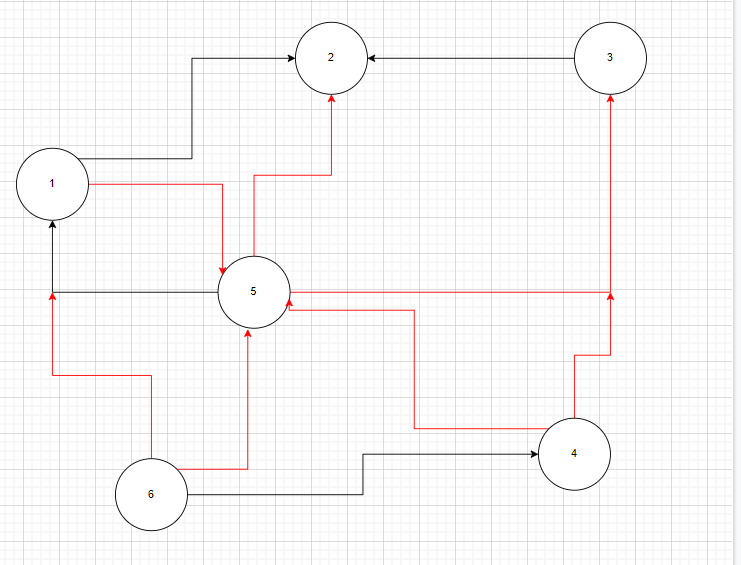
La formula de uso acerca del vecino mas cercano parte de especificar una cantidad numérica de vecinos próximos, teniendo en cuenta que no signifique que por mayor cantidad de vecinos cercanos la calidad del modelo sea mas certera.

Al nosotros especificar un destino variable, podemos especificar que el rango seleccionado por el procedimiento sea escogido por el mejor de la cantidad de vecinos posibles en el rango que se le asigna.

Al asignar esta característica como específica, esta realizara por cada valor de K en un rango solicitado y seleccionara a K y su conjunto de funciones cercanas con una menor tasa de error al momento del escalado.

Al no usar la característica anteriormente mencionada, se validará de mejor manera partiendo de un pliegue entre posibles mejores vecinos y de estos saldrá la comparación del escalado.

Esto usa como base el Cálculo de distancias, las cuales son las métricas que se usa para calcular la distancia empleada entre ambos casos de que un valor sea similar a otro.



**4.2.2 Algoritmo de compresión de imágenes sin pérdida**

LZW (Lempel-Ziv-Welch) es un algoritmo de compresión sin pérdida desarrollado por Terry Welch en 1984 como una versión mejorada del algoritmo LZ78 desarrollado por Abraham Lempel y Jacob Ziv.

Es un algoritmo de compresión adaptativa, lo que quiere decir es que va leyendo el archivo que se desea comprimir y con base en esto va a ajustando lo que vendría siendo el diccionario de datos comprimidos, de tal manera que esta se va generando en la medida que se va leyendo el archivo.

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

Tenemos como complejidad del algoritmo O(N\*M), Esto porque al ser una imagen lo que temenos es una matriz de datos y por eso las variables N y M. Ahora se dice que posee esta complejidad ya que para comprimir por completo el archivo o la imagen debe recorrerla toda.

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

***Tabla* 2:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. *(N serian las filas y M las columnas de la matriz de datos).*

|  |  |
| --- | --- |
| **Algoritmo** | **Complejidad de la memoria** |
| Compresión | O(N\*M ) |
| Descompresión | . |

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes.

**4.4 Criterios de diseño del algoritmo**

Principalmente utilizamos este algoritmo LZW debido a que suele ser mas eficiente que los demás, esto porque para realizar la compresión se hace únicamente con una sola pasada, lo que reduce el tiempo de compresión bastante y aun así tiene muy buena tasa. Otra razón por la que se utilizo es que la mayoría utilizo el LZ77 y por esto decidimos utilizar algo diferente para variar en las presentaciones.A

**5. RESULTADOS**

**5.2 Tiempos de ejecución**

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

|  |  |  |
| --- | --- | --- |
|  | ***Tiempo promedio de ejecución (s)*** | ***Tamaño promedio del archivo (MB)*** |
| *Compresión* | 15m | 1000MB |
| *Descompresión* | . | . |

## **Tabla 6:** Tiempo deejecución del algoritmo *LZW* para diferentes imágenes en el conjunto de datos.

## **5.3 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 | 1MB | 1000MB |
| Descompresión | . | . |

## **Tabla 7:** 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.

|  |  |
| --- | --- |
|  | ***Tasa*** |
| Tasa de compresión promedio | 10:1 |

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

Tamaño inicial/Tamaño final.

## **6. DISCUSIÓN DE LOS RESULTADOS**

Se obtuvieron muy buenos resultados con las pruebas que realizamos, la tasa de compresión fue muy buena, aparte el tiempo de compresión es bastante adecuado para la tamaño de los archivos. También el consumo de memoria es bastante bajo, cosa que nos beneficia bastante y mucho en contexto que manejamos este proyecto. Por ultimo este proyecto, con este modelo de compresión de datos podría mejorar mucho la clasificación de la salud animal en el contexto PFL, ya que haría mucho mas eficiente y efectivo el análisis de estos datos en zonas rurales.

**6.1 Trabajos futuros**

# Lo que creemos que tal vez se podría mejorar a futuro seria el tiempo de compresión de los archivos, aunque este en promedio es bastante bueno.

# **REFERENCIAS**

[1] Rodrigo García, Jose Aguilar, Mauricio Toro, Ángel Pin y Paul Rodríguez. 2020. A systematic literature review on the use of machine learning in precision livestock farming. ELSEVIER 179, 105826. (Dec. 2020) 1 – 12. DOI: https://doi.org/10.1016/j.compag.2020.105826

[2] Vasileios Doulgerakis, Dimitrios Kalyvas, Enkeleda Bocaj, Christos Giannousis, Michalis Fcidakis, George P. Laliotis, Charalampos Patrikakis y Losif Bizelis. 2019. An animal welfare platform for extensive livestock production systems. European Conference on Ambient Intelligence. (Nov. 2019) 1 – 7. CEUR: <http://ceur-ws.org/Vol-2492/paper1.pdf>

[3] Debauche, O., Mahmoudi, S., Andriamandroso, A.L.H. et al. Cloud services integration for farm animals’ behavior studies based on smartphones as activity sensors. J Ambient Intell Human Comput 10, 4651–4662 (2019). <https://doi.org/10.1007/s12652-018-0845-9>

[4] Andrew, W., Greatwood, C., & Burghardt, T. (2018). Visual Localisation and Individual Identification of Holstein Friesian Cattle via Deep Learning. In 2017 IEEE International Conference of Computer Vision Workshop (ICCVW 2017) (pp. 2850-2859). Institute of Electrical and Electronics Engineers (IEEE). <https://doi.org/10.1109/ICCVW.2017.336>

[5] Wikipedia. 2021. Compresión de Borrows-Wheeler. Retrief from <https://es.wikipedia.org/w/index.php?title=Compresi%C3%B3n_de_Burrows-Wheeler&oldid=133166733>

[6] Wikipedia. 2021. Portable Network Graphics. Retrief from <https://es.wikipedia.org/w/index.php?title=Portable_Network_Graphics&oldid=136457907>

[7] Wikipedia. 2021. Codificación Huffman, Retrief from <https://es.wikipedia.org/wiki/Codificaci%C3%B3n_Huffman>

[8] Wikipedia. 2021. LZ77, Retruef from <https://en.wikipedia.org/wiki/LZ77_and_LZ78>

[9] Wikipedia. 2020. Codificación por transformación. Retrief form <https://es.wikipedia.org/w/index.php?title=Codificaci%C3%B3n_por_transformaci%C3%B3n&oldid=127103029>

[10] Wikipedia. 2019. MrSid. Retrief from <https://es.wikipedia.org/w/index.php?title=MrSID&oldid=117890397>

[11] Wikipedia. 2021. Compresión fractal. Retrief from <https://es.wikipedia.org/wiki/Compresi%C3%B3n_fractal>

[12] Wikipedia. 2021. JEPG. Retrief from https://en.wikipedia.org/wiki/JPEG

[13] <https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos>

[14] https://es.wikipedia.org/wiki/Algoritmo\_de\_compresi%C3%B3n\_con\_p%C3%A9rdida#:~:text=Algoritmo%20de%20compresi%C3%B3n%20con%20p%C3%A9rdida%20se%20refiere%20a%20cualquier%20procedimiento,exacta%20de%20los%20datos%20originales.

[15]<https://es.wikipedia.org/wiki/Compresi%C3%B3n_fractal>]

[16] <https://en.wikipedia.org/wiki/Bilinear_interpolation>

[17]<https://www.ibm.com/docs/es/spss-statistics/SaaS?topic=analysis-neighbors-nearest-neighbor>

[18]https://es.wikipedia.org/wiki/LZW

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