Análisis de ganadería de precisión aplicando compresión de imágenes.

Jose Muñoz
Universidad Eafit
Colombia
jmmunozr@eafit.edu.co

Mauricio Correa
Universidad Eafit
Colombia
mdcorreah@eafit.edu.co

Silvia Cáceres
Universidad Eafit
Colombia
skcaceresv@eafit.edu.co

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

RESUMEN

Actualmente la industria ganadera proporciona un tercio de las proteínas consumidas en la dieta humana. Desde hace unos años se presenta un concepto llamado Ganadería de Precisión (GdP), la cual permita unir la tecnología con la ganadería generando así la mejora de esta.

Con base en lo anterior, se diseñó un algoritmo basado en el vecino más cercano y el LZ77, para comprimir y descomprimir imágenes reduciendo el consumo de energía en el contexto ganadero.

El software y la ganadería ya han estado en contacto, gracias a otros estudios con animales de granja: la integración de servicios en la nube para estudios del comportamiento de animales de granja basados en teléfonos inteligentes como sensores de actividad; la localización visual e identificación del ganado Holstein Friesian mediante maching learning; y una plataforma de bienestar animal para los sistemas de producción de ganadería extensiva.

De acuerdo con lo trabajado, se logró generar la compresión y descompresión de imágenes mediante algoritmos con y sin pérdida con una tasa de 4:1. El tiempo y el consumo de ejecución de estos dependerán del tamañode la imagen, sin embargo, se observó que el tiempo de ejecución y el consumo de memoria son de manera exponencial.

Palabras clave

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

1. INTRODUCCIÓN

En 1536 Sebastián de Belalcázar fue uno de los primeros pioneros en traer desde “La Española” actualmente Haití y República Dominicana en traer en ganado bovino a Colombia, esto debido a que la carne en su época venia de especies silvestres que eran originarias del país, las cuales no permitían una adecuada alimentación. Posteriormente se realiza una expansión por los departamentos de la Región Caribe.

La ganadería se veía como una industria prometedora en sectores donde no había minería, sin embargo, no se veía promisoria para las áreas donde existía la minería. En la actualidad colombiana la ganadería es considerada una de las actividades económicas más importantes para el futuro colombiano, ya que está en constante expansión y se espera que a futuro haya exportaciones y supere las calidades del mercado internacional.

Así mismo, la ganadería es fundamental por sus bienes y servicios que presta a la sociedad, como la diversidad de productos de carne, leche, queso, cuero, otras fibras y el uso de los animales para el arado y transporte de carga. Al mismo tiempo, como componente esencial de la agricultura mundial.

Para poder hacer el seguimiento de la salud de los bovinos, se requiere implementar un sistema de vigilancia para el cuidado de ellos. Para esto es requerido el reconocimiento de todos los especímenes, y la mejor forma de hacerlo es por medio de fotografías. Sin embargo, hay que tener en cuenta que van a haber muchas cabezas de ganado, por esta razón se necesitan muchas fotografías que implican gran cantidad de espacio en memoria. Por tanto, teniendo presente lo anterior se requiere la compresión de estas imágenes. Así que por estas razones se procederá a realizar un software que permita realizar la compresión y el posterior análisis de las imágenes.

1.1. Problema

Durante el transcurso de los tiempos, la ganadería y la agricultura se han encargado de abastecer las necesidades de los seres humanos centrándose así en mejorar los precios y la competitividad en estos ámbitos. Sin embargo, en los últimos años se ha presentado un pensamiento más animalista, procurando así que, a pesar de la producción de los alimentos procedentes de los animales, se tenga presente también el respeto al animal como ser que siente. Además, se han presentado estudios que indican que la salud de los consumidores está directamente relacionada con la salud de los animales y un mejor producto para la venta.

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.

Con el fin de lograr este objetivo, es requerido realizar una búsqueda de algoritmos que permitan la compresión de estas imágenes, en un principio se realiza la investigación para un algoritmo con pérdida de información y se decide trabajar con El vecino más cercano. La selección de este algoritmo se toma debido a que se realizó una búsqueda exhaustiva de los

diferentes algoritmos presentados en la entrega uno y fue complejo encontrar información relacionado a estos. Adicional a esto, gracias a que se trabajó y se explicó sobre este algoritmo durante las clases se decide tomar esta decisión.

.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. 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.[1]

2.2. UNA PLATAFORMA DE BIENESTAR PARA SISTEMAS EXTENSIVOS DE PROCUCCIÓN 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. El sistema ha sido desarrollado para el manejo de ovejas de granjas extensivas en la prefectura de Epiro y será validado en dos casos de uso: primero, monitoreo de los principales indicadores de bienestar 6 de las ovejas en un sistema productivo semi extensivo; y segundo, evaluación de las medidas de bienestar del ganado. en un sistema de crianza extensiva.[2]

2.3. INTEGRACIÓN DE SERVICIONS EN LA NUVE 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 fácil mente 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.

Para el artículo, se muestra en conclusión que la nueva arquitectura que ellos presentan es útil para utilizarla de forma portátil, permite la recopilación de datos a alta frecuencia y se adapta fácilmente a muchos casos. Es utilizando en equipos celulares de IPhone, tales como el 5SE, 6S, 7S y 8S, los cuales están calibrados con un nuevo IMU (Unidad de Medición Inercial). Estos equipos son un medio económico para medir el comportamiento del ganado bovino. Sin embargo, se debe de tener presente que, si se utilizan varios dispositivos en simultáneo, va a haber una pérdida de datos. También, se debe considerar buscar otros algoritmos de compresión de datos para optimizar el consumo de energía de la batería. [3]

2.4. 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

El artículo concluye que es posible realizar una identificación del ganado sin necesidad de utilizar métodos de etiquetado existentes en la actualidad; el cual es un proceso intrusivo y dañino para el animal. esto por medio de la utilización de componentes estándar de aprendizaje profundo. Para la identificación, se demostró que las arquitecturas basadas en convolución son muy adecuadas para aprender y distinguir las propiedades del patrón y la estructura dorsal únicos exhibidos por las especies individualmente. El video capturado fue analizado por el LRCN fuera de línea. El cual demostró que, para el artículo en cuestión, fue demostrado que resultar suficiente en los escenarios presentados. [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 sin pérdida

3.2.1 Algoritmo Borrows Wheeler:

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.

Ejemplo:

Entrada	SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.BOXES
Salida	TEXYDST.E.IXIXIXXSSMPPS.B..E.S.EUSFXDIIOIIIT

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. [5]

3.2.2 Algoritmo PNG:

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 mayor profundidad de contraste y otros datos importantes.

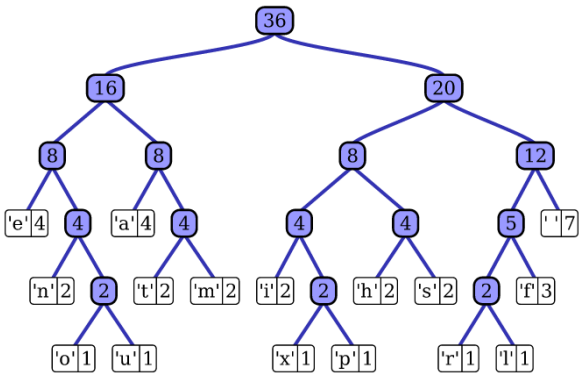
El método de compresión utilizado por el PNG es conocido como deflación (Algoritmo sin perdidas). También tiene métodos de filtrado de información en pixeles, el más utilizado es prediciendo el valor aproximado de los pixeles lo cual mejora la comprensión para cada línea de la imagen a comprimir. En 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. [6]

Profundidad de bits por canal	1	2	4	8	16
Imagen indexada (1 canal)	1	2	4	8	
Escala de grises (1 canal)	1	2	4	8	16
Escala de grises con alfa (2 canales)				16	32
Color verdadero (RGB) (3 canales)				24	48
Color verdadero con alfa (RGBA) (4 canales)				32	64

3.2.3 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 específico para elegir la representación de cada símbolo, que representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa.

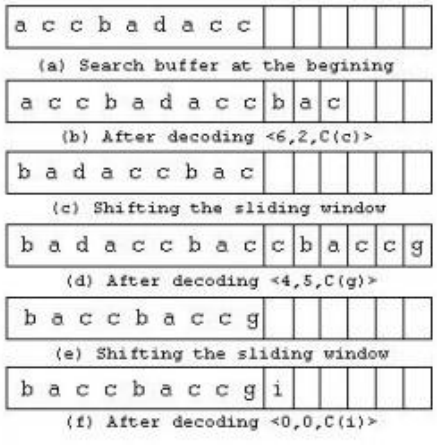


Un ejemplo de un árbol dibujado es el siguiente donde los nodos hijos finales son a los que les corresponde un símbolo y una frecuencia, y los nodos padres de estos últimos se conectan como pequeños árboles a otros nodos. Los nodos con puros números indican la suma de las frecuencias de los nodos hijos correspondientes. [7]

3.2.4 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.[8]



3.3 Alternativas de compresión de imágenes con pérdida

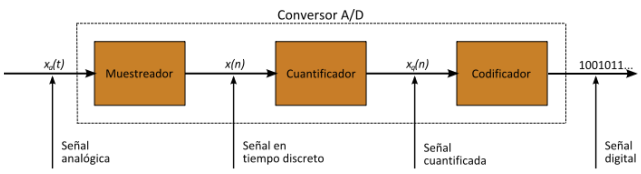
3.3.1Codificación por transformación

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.[9]

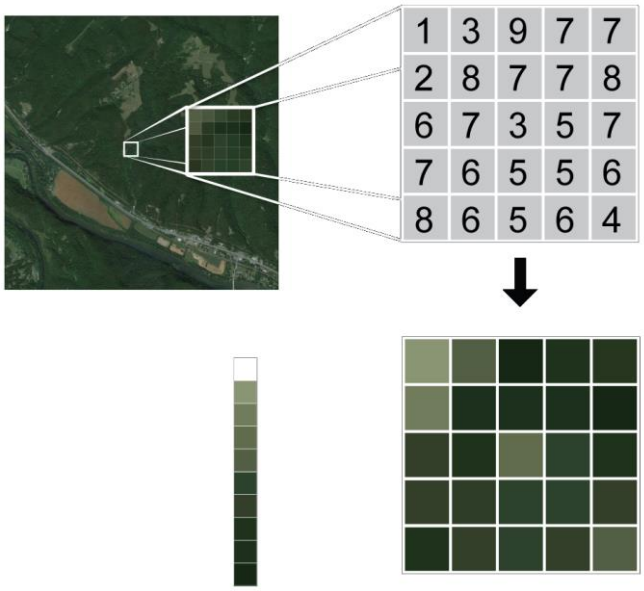


3.3.2 MrSid

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). [10]



3.3.3 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. [11]

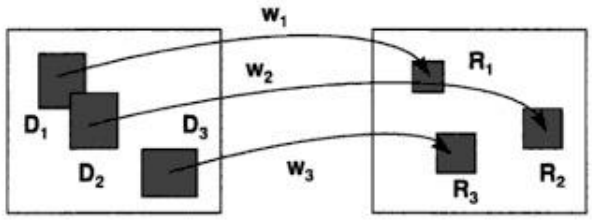


Fig. 3.2.1 A partitioned iterated function system.

3.3.4 Joint Photographic Experts Group (JPEG)

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: [12]

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

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

El vecino más cercano:

El algoritmo del vecino más cercano, es un método que genera el camino más corto de soluciones, pero que por lo general no son las más ideales. Se basa en clasificar los casos buscando la ruta más parecida entre ellos o por lo menos el más cercano a estos. Buscando con base a estos parecidos, la solución más adecuada para un problema. Este método fue diseñado con la intención de reconocer patrones de datos, sin la necesidad de encontrar las coincidencias exactas entre los patrones, conjuntos de datos o casos de almacenamiento.

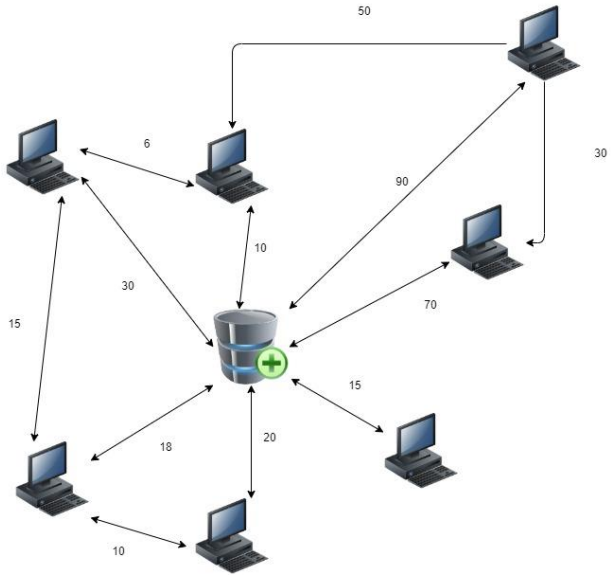


Figura 1: De acuerdo con la figura presentada, requerimos saber cuál es el computador más óptimo para poder operar la

[1]http://www.github.com/ ????????? /proyecto/

base de datos con mayor eficiencia y eficacia. En el grafo presentado en la figura se pretende mostrar cual es el computador más óptimo para esta operación, mientras la línea este más cercana a la base de datos el computador va a ser mejor. Teniendo en cuenta la teoría del vecino más cercano, podemos darnos cuenta que el computador más óptimo para operar la base de datos es el computador más cercano a ella el cual es el que tiene una distancia de 10 unidades, mientras que el peor sería la unión de los computadores que entre ellos tiene una distancia de 30 y entre la base de datos y el computador más cercanos de ellos dos que tiene 70 unidades, siendo así una distancia de 100 unidades.

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.

El código del algoritmo de compresión con pérdidas se está se está realizando en el lenguaje de programación de Python con la versión más nueva de la versión 3. Se toma este lenguaje de programación debido a que se está trabajando durante el semestre lectivo.

Ahora bien, con el fin de poder manipular las imágenes, es requerido utilizar una librería de Python llamada OpenCV, con esta librería podemos hacer el procesamiento de imágenes. Permitiendo así abrir imágenes, cambiarle su tamaño, sobre escribirlas, entre muchas otras funcionalidades más. Gracias a esto, es posible realizar la compresión de imágenes. Sin embargo, con los métodos de cambiarle el tamaño y sobre escribir, se pierde información importante produciendo así el pixelado de estos archivos.

Asimismo, se utiliza otras dos librerías “tkinter” y “OS” para hacer el trabajo más dinámico y poder crear la interfaz gráfica y crear carpetas nuevas sin necesidad de ir a crearlas desde los documentos. Tkinter es la librería para la interfaz gráfica y OS es para poder crear las nuevas carpetas en las rutas especificadas en el código.

Descripción del código.

Inicialmente se importan todas las librerías necesarias para poder realizar sin ningún contratiempo o error todo el código y que funciones a la perfección.

Líneas: 6 a 10

```
import cv2
from tkinter import *
from tkinter import filedialog
import os
```

A continuación, se crea el directorio y el nombre del archivo en la dirección donde se va a implementar, adicionalmente se crea un contador con el fin de que al nombre del archivo de la imagen final vaya cambiando mientras se van creando las imágenes.

Líneas: 11 a 14

```
directorio = filedialog.askdirectory()
nombreDelArchivo = os.listdir(directorio)

contador = 0
```

Posteriormente se crea un ciclo para que vaya recorriendo todos los archivos tipo foto que se encuentren en una carpeta preseleccionada en la ejecución.
Se recibe la dirección de la carpeta, se imprime en la consola cual es la foto que se lee y se guarda en una variable imagenOriginal para posteriormente ser modificada.

Líneas: 15 a 20

```
for lineaDeTexto in nombreDelArchivo:

    caminoAlArchivo = directorio + "/" + lineaDeTexto
    print(nombreDelArchivo)
    imagenOriginal = cv2.imread(caminoAlArchivo)
```

Siguiente a esto se crea la nueva carpeta de y se verifica si existe, si no existe, entonces se crea la nueva carpeta, si sí existe entonces, se reescribe la información interna.

Líneas: 21 a 24.

```
    carpetaNueva = "./carpetaDeImagenes"
    if not os.path.exists(carpetaNueva):
        os.makedirs(carpetaNueva)
```

Ahora bien, la parte más importante del trabajo está en la línea 25. En esta línea encontramos la creación de una nueva imagen (imagenFinal), aquí se realiza el cambio del tamaño de la imagen con el fin de poder hacer la compresión de esta. Para el caso de este código se decidió, con el fin de mostrar el cambio en tamaño, realizar un sobredimensionamiento de la imagen presentada en la figura 2. Y como se puede observar en la parte final de la línea ***“interpolation= cv2.INTER_NEAREST”,*** este algoritmo utiliza la interpolación por medio del método del vecino más cercano.

Línea 25:

imagenFinal = cv2.resize(imagenOriginal, (1920, 1080), interpolation= cv2.INTER_NEAREST)

En las últimas tres líneas se crea la nueva imagen con un nombre prediseñado, se aumenta el contador para poder continuar con las siguientes imágenes y se imprime el nombre de la imagen final.

Líneas: 26 a 28

```
cv2.imwrite(carpetaNueva+"/archivoFinal"+str(contador)
+".jpg", imagenFinal)
```

contador+=1
print("/archivoFinal"+str(contador)+".jpg")

Figura 2: Imagen usada en el código: La imagen de la izquierda es la imagen original "imagenOriginal" y la imagen de la derecha es la imagen habiendo realizado la interpolación por vecino más cercano "imagenFinal"

En un principio se había realizado un trabajo con imágenes de archivos tipo jpg, sin embargo, posteriormente se realiza el análisis utilizando matrices entregadas en archivos csv. En el github se entrega el código final utilizando los algoritmos con perdida: vecino más cercano y el algoritmo sin pérdida: el LZ77. Por ende, el código mencionado anteriormente queda obsoleto.

Imágenes finales del archivo:

Imagen original. Vaca enferma

Imagen con pérdida: Vaca enferma.

4.2.2 Algoritmo de compresión de imágenes sin pérdida

El algoritmo de compresión de imágenes sin pérdida que escogimos fue el LZ77, que es un modelo bastante usado debido a su eficiencia al momento de comprimir información sin admitir perdidas.

Para la compresión de imágenes se debe convertir la matriz de pixeles en un arreglo. El compresor utiliza tripletas donde el primer dígito significa cuantas veces me muevo hacia atrás, el segundo dígito es cuantas letras se toman y el tercer dígito es que letra le sigue a ese grupo. Con este proceso es posible encontrar repeticiones y almacenarlas en las tripletas que contienen toda la información necesaria. Para la descompresión se toman las tripletas obtenidas en la compresión.

4.3 Análisis de la complejidad de los algoritmos

El peor caso para nuestro algoritmo, cuando un píxel es igual al número máximo de número de filas N (número de filas) y el número máximo del número de columnas M (número de columnas). O sea, si tenemos una imagen donde su tamaño es de 1960x1000, el peor caso es cuando el píxel está en la posición N = 1000 y M = 1960. Esto debido a que el algoritmo debe recorrer N = 1000 filas y M = 1960 columnas.

El vecino más cercano	**La complejidad del tiempo**
Compresión	O(N*M)
Descompresión	O(N*M)

***Tabla* 1:** Complejidad temporal del algoritmo del vecino más cercano, de compresión y descompresión de imágenes.

LZ77	**La complejidad del tiempo**
Compresión	O(N*M)
Descompresión	O(N*M)

***Tabla* 2:** Complejidad temporal del algoritmo LZ77, de compresión y descompresión de imágenes.

El vecino más cercano	**Complejidad de la memoria**
Compresión	O(N*M)
Descompresión	O(N*M)

***Tabla* 3:** Complejidad de memoria del algoritmo del vecino más cercano, de compresión y descompresión de imágenes.

LZ77	Complejidad de la memoria
Compresión	O(N*M)
Descompresión	O(N*M)

***Tabla* 4:** Complejidad de memoria del algoritmo LZ77, de compresión y descompresión de imágenes.

Para nuestro algoritmo la complejidad en todo momento es de O(N*M), siendo N el número de filas de las imágenes (Matriz) y M el número de columnas dentro de la matriz luego de la compresión o descomprensión.

4.4 Criterios de diseño del algoritmo

El algoritmo fue diseñado de esta manera debido a que el LZ77 es muy usado por la fácil implementación y su gran eficiencia al momento de comprimir información sin pérdida, este algoritmo lee más rápido los datos a diferencia de otros algoritmos, lo que nos proporciona grandes ventajas en la optimización.

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.

Algoritmo	***Tiempo promedio de ejecución (s)***	***Tamaño promedio del archivo (MB)***
LZ77	17.037 s	0.4 MB
El vecino más cercano	0.09025 s	0.100 MB

Tabla 6: Tiempo de ejecución de los algoritmos

5.3 Consumo de memoria

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

Algoritmo	***Consumo promedio de memoria (MB)***	***Tamaño promedio del archivo (MB)***
LZ77	0.4405 MB	0.4 MB
El vecino más cercano	0.1105 MB	0.100 MB

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.

	Ganado sano	***Ganado enfermo***
Tasa de compresión promedio	4:1	4:1

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

6. DISCUSIÓN DE LOS RESULTADOS

El modelo no está sobre ajustado, ya que en ningún momento se conoce el resultado de los datos para el desarrollo del proyecto. Consideramos que el modelo de consumo de tiempo y memoria son apropiados para los dos algoritmos utilizados, además, podemos darnos cuenta de que para el algoritmo con pérdida son aceptables los datos obtenidos, ya que se espera que la tasa de compresión y tamaño sean muy pequeños teniendo en cuenta la pérdida de la información. Y en cuanto al algoritmo sin pérdida de información, se espera que el tiempo y consumo sean mayores, por lo que también es adecuado los datos obtenidos. La tasa de compresión es la adecuada, ya que al utilizar el algoritmo del vecino más cercano se logra una pérdida de datos sin llegar a afectar la figura ilustrada en las imágenes. Finalmente, esta compresión nos puede ayudar a vigilar de una forma más rápida y controlada los temas relacionados con la salud animal ya que puede ayudarnos a encontrar soluciones óptimas para el cuidado animal y posibles enfermedades de estos.

6.1 Trabajos futuros

En un futuro nos interesa lograr mejorar el algoritmo de tal forma que exista una mayor optimización en la compresión y descompresión.

También nos interesaría alcanzar un mayor aprendizaje en la IA para disminuir los posibles errores que se puedan llegar a tener.

RECONOCIMIENTOS

Esta investigación fue apoyada/parcialmente apoyada por el Gobierno Nacional mediante el programa de Generación E-Componente Excelencia 2020.

El señor Mauricio Correa es apoyado por EAFIT a tu alcance, que le permite estudiar con un préstamo de largo plazo.

Todos los autores agradecemos a la Vicerrectoría de Descubrimiento y Creación, de la Universidad EAFIT, por su apoyo en esta investigación.

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