# **ALGORITMO PARA EL MONITOREO DEL ESTADO DEL GANADO**

# **Algoritmo para de precisión en cuanto al salud de las vacas para tener un buen bienestar animal y unos buenos bovinos.**

|  |  |  |  |
| --- | --- | --- | --- |
| Samuel García Correa  Universidad Eafit  Colombia  sgarciac6@eafit.edu.co | Sebastian Arias Usma  Universidad Eafit  Colombia  sariasu@eafit.edu.co | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

# **RESUMEN**

Se basa en el problema que tiene algunos ganaderos con el monitoreo de su ganado, en cuanto al estado de salud del mismo, aunque es un problema del ámbito ganadero nos incumbe a nosotros ya que del estado del ganado depende la calidad del alimento que consumiremos, como problemas relacionados tenemos, la localización y visualización del ganado, el bienestar animal para producción ganadera extensiva, el comportamiento del ganado.

¿Cuál es el algoritmo propuesto? ¿Qué resultados obtuviste? ¿Cuáles son las conclusiones de este trabajo? El resumen debe tener como máximo **200 palabras**. (*En este semestre, deberías resumir aquí los tiempos de ejecución, el consumo de memoria, la tasa de compresión y la exactitud*).

## **Palabras clave**

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

# **1. INTRODUCCIÓN**

En este proyecto la motivación es el monitoreo de un ganado extensivo y tener una clasificación de su estado de salud, ya que esto también nos afecta, aunque sea del ámbito ganadero, esto puede afectar en nuestra alimentación y en la calidad de nuestros alimentos, decidimos solucionar este problema con un algoritmo el cual comprime imágenes del ganado y clasifica el ganado por su salud en el contexto de la ganadería de precisión.

# **Problema**

En este proyecto nos centramos en la problemática de la clasificación del ganado en el contexto de la ganadería de precisión implementando un algoritmo de compresión de imágenes para saber el estado de salud del ganado, es un problema que abarca muchos sectores como el económico, el alimenticio, el de la salud y el tecnológico, todo esto ya que muchas veces no se puede monitorear de una manera adecuada el ganado y esto afecta nuestra comida y la perdida de ventas de ganado para los ganaderos que se benefician de este mismo, por esta razón hemos decidido resolver el problema con un algoritmo de compresión de imágenes.

**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.

*Expliquen*, brevemente, su solución al problema *(En este semestre, la solución es una implementación de algoritmos de compresión. ¿Qué algoritmos han elegido? ¿Por qué?)*

**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.

## Los trabajos relacionados tratan la investigacion de la salud animal y la compression de datos con perdidas y sin perdidas en el context GpD.

## **3.1 Localización visual e identificación individual del ganado:**

[1] Este problema surge de la identificación y localización individual del Ganado Holstein Fresian de una manera automatica y no intrusiva, el ganado Holstein Freasian es el de mayor producción de leche tipo bovino, por lo tanto se neceista saber la identificación y la trazabilidad de estos bovinos, se han implementados metodos los cuales estan sujetos a perdidas y daños fisicos lo cual pone en riesgo el bienestar animal como lo son:

* Auriculares bovinos
* Tatuajes bovinos

Por lo tanto se ha planeado la identificación visual de los bovinos basada en los patrones del pelaje, la cual contrubuye a la eficancia ganadera y al bienestar animal.

Los algoritmos utilizados fueron R-CNN (Region Based Convolutional Neural Networks), VGG (Very Deep Convolutional Networks for Large-Sacale Image Recogntion) y CNN(Convolutional Neural Network), la arquitectura consta de 5 capas convolucionales apiladas, las cuales se comparten con la red propuesta de la region , más dos capas completamente conectadas, con un modelo entrenado de la base de datos ImageNet suministrado por la implementación en Python Faster-RCNN, la detección y localización de objetos produce regiones de interés de imagenes o en forma de cuadros delimitadores.

[1] Andrew, W., Greatwood, C., & Burghardt, T. (2018). Visual Localisation and Individual Identificationn of Holstein Fresian Cattle via Deep Learning in 2017 IEE International Conference of Computer Vision Workshop (ICCVW 2017) (pp.3850-2859). Institute of Electrical and Electronics Engineers (IEEE), University of Bristol, Bristol UK, department of Computer Science

https://doi.org/10.1109/ICCVW.2017.336

## **3.2 Plataforma de bienestar animal para la ganadería extensiva:**

[1] La producción agrícola se centraba principalmente en las necesidades humanas, el precio y la competencia, pero la evolución en las últimas décadas ha generado que los consumidores esperan que sus alimentos se produzcan de una manera más respetuosa con el bienestar animal, por lo tanto, se ha creado una preocupación mundial mediante al cuidado animal.

Por lo tanto, se presenta un sistema automatizado con un sensor inalámbrico el cual es capaz de registrar los indicadores de bienestar animal como:

* Movimiento.
* Velocidad.
* Información de geolocalización del animal.

Se trabajo sobre algoritmos de reconocimiento de patrones de redes neuronales profundas, la cual proporciona información útil a los agricultores para que tomen las medidas adecuadas para el bienestar animal.

El algoritmo usado fue la arquitectura de una red neuronal convolucional (CNN) la cual está diseñada para procesar los datos sin procesar, además se usan herramientas de procesamiento de fusión de datos para combinar os datos de diferentes sensores que producen nuevos datos sin procesar que pueden ser utilizados como información extra, la cual es guardada en la nube o se puede visualizar en la app del usuario.

[1] Doulgerakis, V., Kalyvas, D., Bocaj, E., Giannousis, C., Fiedakis, M., Laliotis, GP., Patrikakis, C., Bizelis, L. An Animal Welfare Platform for Extensive Livestock Production Systems, University of Wets Africa, Agricultural University of Athens.

## **3.3 El uso del aprendizaje automático en la ganadería de precisión:**

[1] En la ganadería tradicional se crían diferentes especies de animales y se obtienen productos para consumo humano, últimamente ha surgido el término de ganadería de precisión el cual trata de que el proceso de cultivo ganadero sea mejor y eficaz, gracias a esto es posible procesar diariamente los datos relacionados con el control de los animales.

El algoritmo usado clasifica a los animales por su comportamiento en diferentes categorías el cual se considera un mecanismo útil para el seguimiento del campo a largo plazo, el algoritmo clasifica los comportamientos utilizando información brindada por un dispositivo de borde integrado el cual incluye un acelerómetro triaxial y sensores de giroscopio triaxial, otro algoritmo logra la clasificación no supervisada de datos recopilados por unidad de medición inercial (IMU), la cual va en la parte posterior de las vacas lecheras alojadas en un puesto libre.

[1] García, R., Aguilar, J., Toro, M., Pinto, A., Rodríguez, P. (2020). A systematic literature review on the use of machine learning in precision livestock farming. Universidad del Sinú, Universidad de los andes Mérida, Universidad Eafit.

## **3.4 Construyendo representaciones digitales para acercar a los animales hacia el agricultor:**

[1] La mayoría de agricultores se encuentran en una situación la cual para que ellos mantengan su sustento deben explorar las economías a escala, como resultado ellos tienen un tiempo limitado para interactuar con sus animales, mientras que la atención al bienestar animal ha aumentado y el sector ganadero es mas consciente de esta gestión, la sociedad exige más atención a las necesidades de cada animal.

Por esto la evolución de las tecnologías de la información y la comunicación ha tenido un impacto en la agricultura, ya que en los últimos años han sido testigos de la explosión de nuevos sistemas de software, dispositivos de monitoreo y maquinas que explotan las ultimas capacidades de detección, comunicación y procesamiento. Estas funciones optimizadas representan los procesos agrícolas y buscan interactuar con los propios animales, el animal es la parte central del proceso y la tecnología debe apoyarlos para lograr una vida mejor para ellos y para nosotros.

[1] Norton, T., Chen, C., Larsen, M.L.V., Berckmans, D. (2019). Review: Precision livestock farming: biuilging ‘digital representations’ to bring the animals closer to the Farmer. Aarthusr University.

## **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 Seam Carving**

Es un algoritmo usado para reorientación de imágenes, funciona estableciendo uniones (pueden ser horizontales o verticales) en una imagen y las elimina para disminuir el tamaño de la imagen o inserta para agrandar la imagen. Este cambio de tamaño puede generar distorsiones en los objetos que hay en la imagen. Este algoritmo trabaja con la importancia de los píxeles y se contrasta con los valores de los píxeles vecinos dependiendo su intensidad y energía, siendo que los que poseen menores valores serán de menor importancia y los que tiene más valores será de mayor importancia.

El proceso de funcionamiento del algoritmo es el el siguiente:

Comienza tomando una imagen cualquiera, luego se calcula el peso / densidad / energía de cada pixel, y a partir de la energía se hará una lista de costuras o uniones que se clasificaron por energía, luego se quitan las costuras de baja energía y ya con esto nos quedará la imagen final.

[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://en.wikipedia.org/wiki/Seam\_carving

Imagen original:



Imagen final:



**3.2.2 Image scaling**

El escalado de imágenes es el cambio de tamaño o resolución de una imagen digital, al escalar una imagen de gráfico vectorial , las gráficas primitivas de una imagen se pueden escalar utilizando transformaciones geométricas, sin perder calidad de imagen. Esto generando una nueva imagen con un número mayor o menor de píxeles. Cuando queremos reducir la resolución de una imagen primero debemos aplicar un filtro de antialiasing, lo cual es una tecnología que nos permite quitar los picos o cuadros que se pueden generar cuando se baja la calidad de una imagen. Existen diferentes algoritmos que nos permiten cambiar el tamaño o resolución de las imágenes de diferentes formas pero generalmente lo que hacen es crear nuevas imágenes creando más o menos pixels que los que tenía la imagen original, esto por medio de diferentes técnicas y cálculos matemáticos, la siguiente imagen fue mejorada utilizando diferentes algoritmos:

[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://en.wikipedia.org/wiki/Image\_scaling

**3.2.3 Transformación de coseno discreto**

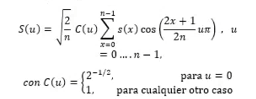
Es un algoritmo altamente usado en el procesamiento de señales y la compresión de datos. Además se utiliza en la mayoría de los medios digitales, como imágenes, video, audio, televisión, radio y codificación de voz. Este algoritmo usa funciones de coseno en lugar de seno ya que es más eficiente para la compression, la fórmula es la siguiente:



Con la formula de Euler seria así:



Si tenemos una lista S(x) de números hasta n. entonces la transformada de cosenos discreta es la siguiente:



En la compresión de imágenes con pérdida se necesita esta forma en dos dimensiones ya que necesitamos tener una matriz de píxeles.

Una imagen contiene pixeles que son números reales dentro de una matriz de ancho x largo siendo S(u,v)2 la magnitud del pixel, por lo que se usa esta fórmula:

Lo que ocurre en esta transformación con relación a los pixeles de la matriz es que cada uno de los pixeles originales (s(x,y)) cambia con relacion a los pixeles contiguos a ellos en la nueva matriz creada por la transformada.

**La complejidad del algoritmo es: O(N2 log2 N)**

[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://es.wikipedia.org/wiki/Transformada\_de\_coseno\_discreta

**3.2.4 Compresión fractal**

Es un metodo de compresión de imagenes que se basa en el hecho de que algunas partes de la imagen se parecen a otra de la misma imagen y esas partes se usan para recrear la imagen codificada.

Para una imagen binaria donde la imagen es un subconjunto de R2 . SFI es un conjunto de contracciones

****

De acuerdo con estas funciones, el SFI describe un conjunto bi-dimensional S como el punto fijo del operador de Hutchinson

****

Siendo, H un conjunto operador a conjunto, y S el único conjunto que satisface H(S) = S. La idea es construir el SFI de tal forma que el conjunto S es la imagen binaria. El conjunto S puede ser recuperado del SFI mediante el método del punto fijo: para cualquier conjunto inicial compacto no vacío A0, la iteración Ak+1 = H(Ak) converge a S.

El conjunto S es igual a sí mismo porque H(S) = S implica que S es una unión de copias de sí mismo:



Así que vemos que SFI es una representación fractal de S.

SFI también se puede dar para una imagen en escala de grises pero la gráfica de la imagen sera un subconjunto de R3 Para una imagen en escala de grises u(x,y), considerar el conjunto S = {(x,y,u(x,y))}. Entonces, al igual que el caso binario, S se describe por un SFI utilizando una serie de contracciones ƒ1,...,ƒN, pero en R3



[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://es.wikipedia.org/wiki/Compresi%C3%B3n\_fractal

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

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

**3.3.1 Codificación aritmética**

Es un algoritmo de compresión sin pérdida que convierte de bit a codificación aritmética, a diferencia de otras formas de codificación, la aritmética codifica el mensaje entero a un solo número.

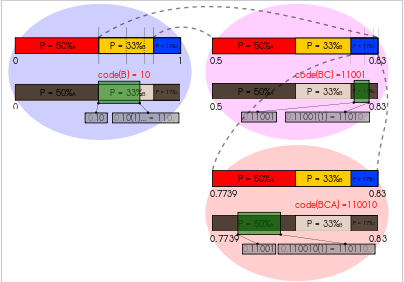
Los codificadores aritméticos suelen producir una salida cerca del óptimo para cualquier conjunto de símbolos y probabilidades dado. Los algoritmos de compresión que usan la codificación aritmética inician determinando un modelo de los datos básicamente una predicción de que patrones serán encontrados en los símbolos del mensaje. Lo más acertada que sea la predicción, lo más cerca al óptimo que sea la salida.

En general, cada paso del proceso de codificación, excepto por el último, es lo mismo; el codificador tiene sólo tres piezas de datos a considerar:

1. El siguiente símbolo que necesita ser codificado.
2. El intervalo actual (al inicio del proceso de codificación, el intervalo es [0,1], pero eso cambiará).
3. Las probabilidades que el modelo asigna a cada uno de los varios símbolos que son posibles en esta etapa

El codificador divide el intervalo actual en sub-intervalos, cada uno representando una fracción del actual intervalo proporcional a la probabilidad de ese símbolo en el contexto actual. Cualquiera que sea el intervalo que corresponda al símbolo actual que sigue a ser codificado se vuelve el intervalo usado en el siguiente paso.

[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://es.wikipedia.org/wiki/Codificaci%C3%B3n\_aritm%C3%A9tica

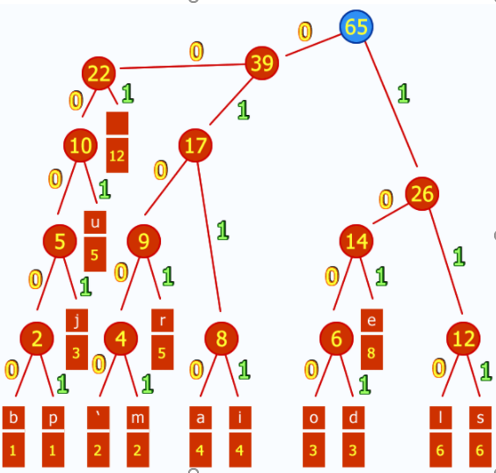


**3.3.2 Codificación de Huffman**

Es un algoritmo usado para la compresión de datos y usa un método para elegir la representación de cada símbolo, que da lugar a un código prefijo que representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa. La técnica básica del algoritmo de Huffman es la creación del árbol binario en el que se etiquetan los nodos hoja con los caracteres, junto a sus frecuencias, y de forma consecutiva se van uniendo cada pareja de nodos que menos frecuencia sumen, pasando a crear un nuevo nodo intermedio etiquetado con dicha suma. Se procede a realizar esta acción hasta que no quedan nodos hoja por unir a ningún nodo superior, y se ha formado el árbol binario.

Posteriormente se etiquetan las aristas que unen cada uno de los nodos con ceros y unos. El código resultante para cada carácter es la lectura, siguiendo la rama, desde la raíz hacia cada carácter (o viceversa) de cada una de las etiquetas de las aristas. El tamaño de los arboles depende del número de símbolos, n, donde los nodos contienen dos campos, el símbolo y el peso. El proceso de construcción del árbol comienza formando un nodo intermedio que agrupa a los dos nodos hoja que tienen menor peso (frecuencia de aparición). El nuevo nodo intermedio tendrá como nodos hijos a estos dos nodos hoja y su campo peso será igual a la suma de los pesos de los nodos hijos. Los dos nodos hijos se eliminan de la lista de nodos, sustituyéndolos por el nuevo nodo intermedio. El proceso se repite hasta que solo quede un nodo en la lista. Este último nodo se convierte en el nodo raíz del árbol de Huffman.

[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://es.wikipedia.org/wiki/Codificaci%C3%B3n\_Huffman#:~:text=En%20ciencias%20de%20la%20computaci%C3%B3n,usado%20para%20compresi%C3%B3n%20de%20datos.&text=Aunque%20la%20codificaci%C3%B3n%20de%20Huffman,veces%20puede%20verse%20accidentalmente%20exagerada.

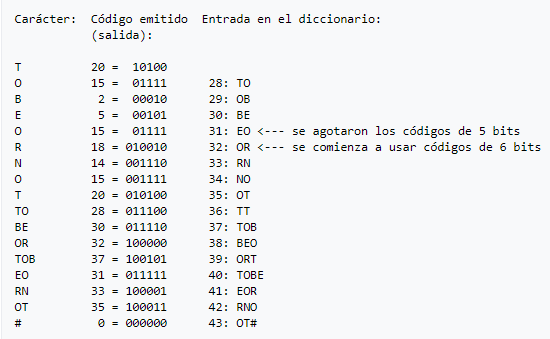


**3.3.3 algoritmo LZM**

es un algoritmo de compresión sin pérdida, que a diferencia de otros métodos de compresión es posible crear sobre la marcha, de manera automática y en una única pasada un diccionario de cadenas que se encuentren dentro del texto a comprimir mientras al mismo tiempo se procede a su codificación.

El método llegó a ser utilizado de forma moderada, pero en toda su amplitud en el programa compress que llegó a ser más o menos la utilidad estándar de compresión en sistemas Unix alrededor de 1986. Otras utilidades de compresión también utilizan este método u otros relativamente cercanos. Se usó ampliamente desde que se convirtió en parte del formato gráfico GIF en 1987. Puede ser también usado, en archivos TIFF.

La compresión LZW proporcionaba una relación de compresión mejor en muchas aplicaciones que otros métodos de compresión conocidos en esa época. Llegó a convertirse en el primer método de propósito general de compresión de datos usado ampliamente. En textos largos, comprime aproximadamente a la mitad del tamaño original. Otros tipos de datos son también comprimidos útilmente en muchos casos.

Un ejemplo de este algoritmo es el siguiente: se supone que los textos a comprimir se componen solamente de letras mayúsculas sin espacios, para lo cual bastan (en inglés) 26 códigos, del 1 al 26, para las mayúsculas más un código para representar el fin de archivo, que se ha representado gráficamente por el símbolo #. El texto a comprimir es “TOBEORNOTTOBEORTOBEORNOT#”. El proceso de compresión queda representado por la tabla siguiente: El texto original, compuesto de 25 caracteres pueden representarse con 5 bits cada uno nos daría 125 bits. El resultado comprimido produce 5 códigos de 5 bits más 12 códigos de 6 bits, lo cual resulta en 97 bits, una reducción a menos del 78% del original. Nótese que cada carácter leído genera una nueva entrada en el diccionario, independientemente de si se utilizará o no. Esta simplicidad por parte del algoritmo de compresión permite que el descompresor pueda reconstruir el diccionario sin errores.

[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://es.wikipedia.org/wiki/LZW

**3.3.4 Compresión de Burrows-Wheeler**

Es un algoritmo de compresión que Cuando se transforma una cadena de caracteres mediante la BWT, ninguno de sus caracteres cambia de valor. La transformación permuta el orden de los caracteres. Si la cadena original contiene muchas subcadenas que aparecen a menudo, entonces la cadena transformada contendrá múltiples posiciones en las que un mismo carácter esté repetido varias veces en una fila

Por ejemplo:



## La transformación se realiza ordenando todas las rotaciones del texto en orden lexicográfico, y seleccionando la última columna. Por ejemplo, el texto "^BANANA@" se transforma en "BNN^AA@A" a través de estos pasos (el símbolo rojo @ indica el 'EOF (final de fichero)'):



Para entender por qué se crean datos más fáciles de comprimir, se puede considerar la transformación de un texto largo en inglés que contenga frecuentemente la palabra "the". Ordenando las rotaciones de este texto a menudo agrupará rotaciones que empiecen por "he ", y el último carácter de esa rotación normalmente será "t", por lo que el resultado de la transformación contendrá un número de caracteres "t" junto con algunas excepciones menos comunes. Por lo tanto, se puede ver que el éxito de esta transformación depende de un único valor con una probabilidad muy alta de ocurrencia antes de una secuencia, por lo que en general necesita muestras muy largas de datos apropiados.

Lo más interesante sobre la BWT no es que genere una salida más fácil de codificar, una ordenación ordinaria podría hacerlo igualmente, sino que es un proceso reversible, permitiendo re-generar el documento original a partir de la última columna de datos.

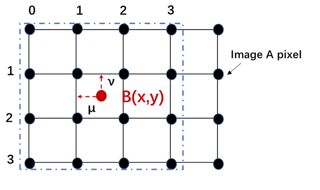
[3] Wikipedia. 2021. Wikipedia: la enciclopedia libre. Obtenido de: https://es.wikipedia.org/wiki/Compresi%C3%B3n\_de\_Burrows-Wheeler

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

## En lo que sigue, explicamos las estructuras de datos y los algoritmos utilizados en este trabajo. Las implementaciones de las estructuras de datos y los algoritmos están disponibles en Github.

## **4.1 Estructuras de datos**

La estructura de datos que nosotros utilizamos para hacer la compresion de imagenes con Perdida fue la matriz



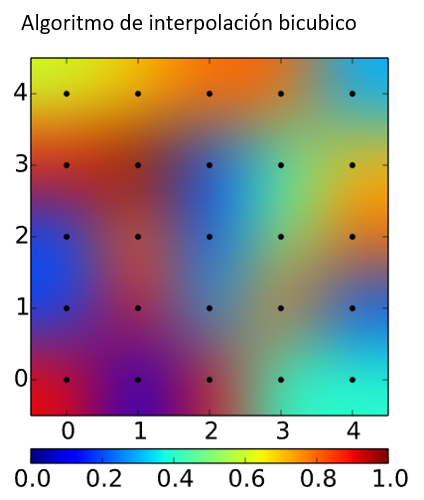
**Figura 1:** Dada una imagen se trata de tomar los valores de un punto en B (x, y) en una cuadrícula e interpolarlos para aproximar el valor de su punto circundante.

**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(bicubic interpolation**

Para esta entrega nosotros utilizamos el algoritmo de compresión de imágenes con Pérdida “bicubic interpolation” el cual se implementó importando python algunas librerías y creando unos cuantos métodos. Este algoritmo es de interpolación no adaptativa y se suele usar en muchos programas de edición de imágenes. Lo que hace este algoritmo es estimar datos desconocidos a partir de datos conocidos. Por ejemplo, si tenemos una imagen con una DSLR a 16 MP, entonces sus datos conocidos serían tener 4928 x 3264 píxeles. En el caso de querer aumentar la escala de la imagen, se aproximan los nuevos valores en función de los píxeles circundantes. Por ejemplo, si desea aumentar el tamaño de la imagen a 24 MP a 6000 x 4000 píxeles, está agregando más píxeles que no estaban presentes antes.



**Figura 2:** la figura anterior es un cuadrado (4,0) x (0,4) y cada cuadrado representa un píxel. Tiene un total de 25 píxeles . Los puntos negros representan los datos que se están interpolando, que suman 25 puntos. Los colores indican valores de función, por lo que en este ejemplo vemos que no son radialmente simétricos. Esto permite un remuestreo más suave con pequeños artefactos de imagen

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

Explique brevemente cómo aplicó un algoritmo de compresión de imágenes sin pérdidas. Para la compresion y descompresion de imagenes sin Perdida utilizamos el algoritmo Huffman, el cual tiene como idea principal asignar códigos de longitud variable a los caracteres de entrada.

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

Explique, con sus propias palabras, el análisis del peor caso usando la notación O. ¿Cómo calculó tales complejidades. Por favor, explique brevemente.

|  |  |
| --- | --- |
| **Algoritmo** | **La complejidad del tiempo** |
| Compresión | O(nlogn) |
| Descompresión | O(nlogn) |

***Tabla* 2:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. Esta complejidad se da ya que hay una cantidad de O(n) interacciones para cada elemento

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

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. Esta complejidad se da ya que hay una cantidad de O(n) interacciones para cada elemento

**4.4 Criterios de diseño del algoritmo**

Nosotros Elegimos este algoritmo Hoffman ya que nos resulto mas sencillo poder tener tanto la descompresion como la compression en el mismo algoritmo, de esta forma el consumo de memoria y tiempo serian iguales tanto para la compression como para la descompresion. Ademas de darnos menos problemas a la hora de ejecutar el codigo y ser un algoritmo eficiente

**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.

Calcular el tiempo de ejecución de cada imagen en Github. Informar del tiempo medio de ejecución vs. el tamaño medio del archivo.

## 

|  |  |  |
| --- | --- | --- |
|  | ***Tiempo promedio de ejecución (s)*** | ***Tamaño promedio del archivo (MB)*** |
| *Compresión* | 15.2 s | 5.2 MB |
| *Descompresión* | 15.1 s | 5.2 MB |

## **Tabla 6:** Tiempo deejecución del algoritmo Huffmanpara la compresion y descompresion de 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 | 50 MB | 5.2 MB |
| Descompresión | 50 MB | 5.2 MB |

## **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.

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

## **Tabla 8:** La Tasa de compresión promedio para el ganado sano y el ganado enfermo es de 4:1, ya que lo tomamos con el algoritmo de compresion con perdidas el cual trabaja con el vecino mas cercano.

## **6. DISCUSIÓN DE LOS RESULTADOS**

En el proyecto realizado para la ganadería de precisión hemos observado diferentes escenarios con diferentes algoritmos los cuales sirven para la compresión de imagenes, primero utilizamos la interpolación bicubica con un algoritmo el cual no era lo mas eficiente pero lograba un muy buen resultado final, en cuanto a tiempo cada imagen tenia un promedio de 7 segundos de compresión este tiempo variando 1 o 2 segundos mas dependiendo de su tamaño y reduciendo tamaño de memoria de la imagen a mas de la mitad de la original, luego tenemos el Huffman coding el cual se encarga de comprimir sin perdidas y descomprimir, este trabaja con archivos .csv y se pasa a un archivo .bin el cual tiene la mitad de tamaño en memoria que el original, tiene un promedio de tiempo de 15 segundos con imagenes de 2 hasta 5 mb lo cual lo hace muy eficiente, Esto puede ser una gran ayuda para las ganadería de precisión pues no presentamos un uso significativo de memoria al usar el programa ya que su promedio estuvo entre las 20mb y las 140 mb por lo que es eficiente al trabajarlo en espacios con computadoras limitadas y de poca capacidad.

**6.1 Trabajos futuros**

Responda ¿qué le gustaría mejorar en el futuro? ¿Cómo le gustaría mejorar su algoritmo y su implementación? ¿Qué tal usar la transformación de coseno discreto o la compresión con ondeletas a futuro?

Nos gustaría tener un algoritmo que sea mas eficiente que el que tenemos actualmente, deforma que no tarde tanto tiempo en comprimir las imágenes y a su vez que conserven su calidad original

# **RECONOCIMIENTOS**

Agradecemos el apoyo suministrado por nuestros compañeros de carrera, nuestro profesor y monitores en el desarrollo del proyecto, gracias a ellos fue posible llevar a cabo este proyecto

# **REFERENCIAS**

1.Adobe Acrobat Reader 7, Asegúrate de que el texto de las secciones de referencia es *Ragged Right*, Not Justified. http://www.adobe.com/products/acrobat/.

2. Fischer, G. y Nakakoji, K. Amplificando la creatividad de los diseñadores con entornos de diseño orientados al dominio. en Dartnall, T. ed. Artificial Intelligence and Creativity: An Interdisciplinary Approach, Kluwer Academic Publishers, Dordrecht, 1994, 343-364.