IDENTIFICACIÓN Y CLASIFICACIÓN DE SALUD ANIMAL EN LA GANADERÍA DE PRECISIÓN

|  |  |  |  |
| --- | --- | --- | --- |
|  | Samuel Botero Mejía  Universidad EAFIT  Colombia  sboterom2@eafit.edu.co | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

# **RESUMEN**

El problema se centra en la salud animal y en un algoritmo de clasificación que permita identificar si estos se encuentran sanos o no. este, para realizar estas clasificaciones, usa fotografías del ganado, es indispensable que las imágenes sean comprimidas de tal forma que la exactitud de la clasificación no se vea afectada en más de un 5%. Otro de sus objetivos es optimizar el consumo de energía y tiempo debido a que este será usado en zonas rurales donde la energía es mas costosa. Este problema es importante porque mejora la calidad de vida del ganado y la rentabilidad de los ganaderos. Este tipo de algoritmos de clasificación se usan a menudo en la ganadería de precisión para estudiar el comportamiento animal y la diferenciación entre razas de ganado. Para la solución de esta se implementaron 2 algoritmos de compresión de imágenes tanto con perdida como sin perdida, el algoritmo con perdida es la interpolación o escalamiento de imágenes por el vecino mas cercano, y en el caso de sin perdidas el código de huffman.

## **Palabras clave**

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

# **1. INTRODUCCIÓN**

La ganadería de precisión usa diferentes tecnologías para recolectar datos, estos deben ser usados de una manera eficiente, y para esto se usan algoritmos de compresión, en este caso serán usados con imágenes, los cuales disminuyen el consumo de energía en los procesos de GDP.

# **1.1. Problema**

Teniendo en cuenta que la ganadería es un proceso vital para la producción de alimentos humanos, es importante que este proceso se controlado y optimizado, para esto sehace uso de la ganadería de precisión la cual permite recolectar datos de manera remota analizar información importante que determinara como se está llevando la producción y así evitar impactos negativos en el ambiente, la salud animal y la utilidad

**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 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.

Mejorar la producción sin consecuencias ambientales o de salud para el ganado

**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** Identificación de ganado Holstein

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

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

Referencia:

Andrew,W., Greatwood, C., Burghardt, T. Visual Localisation and Individual Identification of Holstein Friesian Cattle via Deep Learning. IEEE International Conference of Computer Vision Workshop (2017), ICCVW, 2850-2859.

**3.2** Compresión de imágenes en videojuegos

El contenido se divide en:

1.Por qué las consolas de videojuegos antiguas se ven mal en televisores modernos?

2.compresión de imagen aplicada en la transmisión de juegos.

1.Resolucion SNES de 256×224 del videojuego

1920×1080, HD de definición estándar del TV.

Otros asuntos

-Efecto de fósforo de floración

-Escaneo de líneas

-las señales de 240p a veces ni siquiera son compatibles con televisores modernos

-la baja resolución no afectó a los televisores CRT

Soluciones.

1. Descargas digitales .

2. Manejar la ampliación en la consola de videojuegos.

3. Emulación.

Juegos en la nube: enfoque de transmisión de datos

La ejecución del juego ocurre en los servidores, y los comandos gráficos se capturan y transmiten a los clientes para la representación de la pantalla. Su principal limitación es que consume mucho ancho de banda.

Solución: Live render

Reduce el ancho de banda de la transmisión al comprimir datos gráficos y usar caches. El almacenamiento de cache está dirigido a reducir redundante transmisión mediante la sustitución de estructuras de datos reutilizables con referencias cortas.

Alternativas

1.Xbox Game pass

2.PlayStation Now

3.GeForce Now

4.google stadia

5.Shadow

Referencia:

Marulanda, JM. Image compression in video games. 19.

## **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 Nombre del primer algoritmo**

**Seam carving**

El Seam Carving primero establece un determinado número de costuras (en inglés, seams) a partir de la imagen original. Dichos seams son las partes con la menor relevancia en la imagen y por lo tanto las que deberán ser eliminadas o añadidas (carving) en el proceso final , no obstante este algoritmo también permite la elección manual de las áreas de interés que no deben ser modificadas debido a su importancia.

**3.2.2 Compresión fractal**

La compresión fractal es un método de compresión con pérdida de imágenes digitales, basado en fractales. se basa 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" que se utilizan para recrear la imagen codificada. Con la compresión fractal, la codificación muy costosa desde el punto de vista computacional debido a la búsqueda utilizada para encontrar las auto-similitudes. Sin embargo, la decodificación es bastante rápida.

**3.2.3** Algoritmo JPEG

Es un algoritmo de compresión con pérdidas que aprovecha las deficiencias del ojo humano para comprimir la imagen de forma que aun pueda distinguirse la figura, principalmente se recomienda usar para imágenes con bloques de colores amplios ya que a medida que se comprime, la imagen va perdiendo un poco más la calidad original. Este método de compresión agrupa un conjunto de pixeles continuos que tengan tonos de color similares y los organiza en bloques.

En resumen, el algoritmo divide las imágenes en trozos de 8x8 pixeles y se le aplica la transformada del coseno en dos dimensiones que permite construir la imagen en ondas coseno superpuestas. Luego ajusta los valores de los coeficientes de la imagen y se dividen entre una tabla de cuantización. Finalmente, la unión de las ondas y la tabla resultante nos da una imagen muy parecida a la original, pero con pérdida de calidad.

 

**3.2.4** Algoritmo Live Render

Este algoritmo es usado para consumir menor ancho de bando cuando usamos transmisión de gráficos desde los servidores a nuestro computador, lo que hace es sacrificar la renderización de la imagen a cambio de usar menos ancho de banda. El almacenamiento en cache busca reducir información repetida a través de la sustitución de estructuras de datos con referencias cortas.

**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 Quad tree para imágenes**

Los quadtrees son unas estructuras jerárquicas que usan la recursividad para descomponer espacios, el nombre se quad se refiere a que cada nodo se divide en 4 sub espacios llamados childs; en imágenes se van dividiendo en 4 cuadrantes hasta llegar al nivel de 1pixel, para que la compresión si sea efectiva se mira si el color de los cuadrantes hijos son similares y en caso de que si se toma el color medio del nodo anterior, convirtiendo cierta cantidad de pixeles en solo uno de esta manera disminuyendo el número de nodos y comprimiendo mejor la imagen, es importante que la imagen este en potencias de 2.

**3.3.2 Nombre del segundo algoritmo**

Por favor, explique el algoritmo, su complejidad e incluya una figura vectorial.

**3.3.3** Algoritmo de Huffman

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

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

Se repiten los pasos hasta terminar con todos los caracteres.

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

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

Diagrama

Descripción generada automáticamente

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

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

a b r a c a d a b r a

Esto representa un ahorro del 75%

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

**3.3.4** Algoritmo LZ77

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

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

Antes hay que tener claros algunos conceptos que usaremos

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

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

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

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

Calendario

Descripción generada automáticamente con confianza baja

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

## En lo que sigue, explicamos las estructuras de datos y los algoritmos utilizados en este trabajo. Las implementaciones de las estructuras de datos y los algoritmos están disponibles en Github[[1]](#footnote-1).

## **4.1 Estructuras de datos**

Para el algoritmo de compresión sin perdidas se uso un árbol binario

**Figura 1:** Árbolde Huffman generado a partir de las frecuencias exactas del texto "this". (Por favor, no dude en cambiar esta figura si utilizan una estructura de datos diferente).

**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.

Expliquen el diseño de los algoritmos para resolver el problema y hagan una figura. No uses figuras de Internet, haz las tuyas propias. *(En este semestre, un algoritmo debe ser un algoritmo de compresión de imágenes con pérdidas, como el escalado de imágenes, el tallado de costuras o la compresión con ondeletas, y el segundo algoritmo debe ser un algoritmo de compresión de imágenes sin pérdidas, como la codificación Huffman, LZS o LZ77).*

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

Explique, brevemente, cómo se aplicó un algoritmo de compresión de imágenes con pérdidas, como, por ejemplo, el tallado de costuras o el escalado de imágenes. Explique también la descompresión.

**Figura 2:** Escaladode la imagen mediante interpolación bilineal. (Por favor, siéntase libre de cambiar esta figura si utiliza una estructura de datos diferente).

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

Explique brevemente cómo aplicó un algoritmo de compresión de imágenes sin pérdidas como la codificación Huffman, LZS o LZ77. Explique también la descompresión.

**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(N2\*M2) |
| Descompresión | O(N3\*M\*2N) |

***Tabla* 2:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. *(Por favor, explique qué significan N y M en este problema).*

|  |  |
| --- | --- |
| **Algoritmo** | **Complejidad de la memoria** |
| Compresión | O(N\*M\*2N ) |
| Descompresión | O(2M\*2N) |

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. *(Por favor, explique qué significan N y M en este problema).*

# **REFERENCIAS**

<https://www.youtube.com/watch?v=vGNoaOqyTGY>

<http://www.kramirez.net/RI/Material/Internet/T3_CODIGO_DE_HUFFMAN.pdf>

<https://towardsdatascience.com/huffman-encoding-python-implementation-8448c3654328>

<https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>

<https://www.101computing.net/lossless-compression-huffman-coding-algorithm/>

<http://sedici.unlp.edu.ar/bitstream/handle/10915/2148/Documento_completo.pdf?sequence=1&isAllowed=y>

https://tech-algorithm.com/articles/nearest-neighbor-image-scaling/#:~:text=Nearest%20neighbor%20is%20the%20simplest,%2C%20sinc%2C%20and%20many%20others.

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