

**ESTRUCTURA DE DATOS PARA EL BIENESTAR ANIMAL**

*Proyecto centrado en la implementación de métodos de compresión en el ámbito ganadero para hacerle frente al problema de atención a los animales más vulnerables.*



LUIS MIGUEL MARTINEZ GONZALES

Universidad EAFIT

Colombia

lmmartineg@eafit.edu.co

VICTOR MANUEL BOTERO GOMEZ

Universidad EAFIT

Colombia

vmboterog@eafit.edu.co

14 de agosto de 2021

**CONTENIDO**

[1. INTRODUCCION 2](#_Toc80012344)

[2. PROBLEMA 2](#_Toc80012345)

[3. RESUMEN: 2](#_Toc80012346)

[4. PROBLEMAS RELACIONADOS 3](#_Toc80012347)

[4.1. UNA PLATAFORMA DE BIENESTAR ANIMAL PARA LA GANADERÍA EXTENSIVA 3](#_Toc80012348)

[4.2. VISUAL LOCALISATION AND INDIVIDUAL IDENTIFICATION OF HOLSTEIN FRIESIAN CATTLE VIA DEEP LEARNING: 4](#_Toc80012349)

[4.3. GANADERÍA DE PRECISIÓN EN VACUNO DE CARNE: 5](#_Toc80012350)

[5. ALTERNATIVAS DE SOLUCION 7](#_Toc80012351)

[5.1. ALGORITMO DE HUFFMAN 7](#_Toc80012352)

[5.2. TRANSFORMADA DE BURRORWS-WHEELER: 9](#_Toc80012353)

[5.3. CODIFICACIÓN DE LONGITUD DE EJECUCIÓN (RLE) 10](#_Toc80012354)

[5.4. CODIFICACIÓN ARITMÉTICA 12](#_Toc80012355)

[6. ALGORITMOS CON PERDIDA 13](#_Toc80012356)

[6.1. LA TRANSFORMADA DISCRETA DEL COSENO (DCT) 13](#_Toc80012357)

[6.2. JEPG 15](#_Toc80012358)

[6.3. MP3 15](#_Toc80012359)

[6.4. COMPRESIÓN FRACTAL 16](#_Toc80012360)

[7. REFERENCIAS: 18](#_Toc80012361)

# INTRODUCCION

En la actualidad, el desarrollo de la tecnología ha tomado un papel fundamental en la resolución de problemas de toda índole, esto con el fin de facilitar las labores que el hombre desempeña de una manera óptima, practica y eficaz. Un gran ejemplo de lo mencionado ha sido el problema en el sector agrario con respecto a la identificación del ganado enfermo, que en la mayoría de los casos se identifica con analizar su aspecto físico, ojos, orejas y hocico. El tener un algoritmo que realice este trabajo de clasificación dosifica el trabajo humano y hace más eficiente el proceso respecto al factor tiempo para que el ganado pueda recibir las atenciones necesarias con el fin de preservar la buena salud y vitalidad, es fundamental el papel que toma un algoritmo de clasificación de esta magnitud ya que esto se traduce en disminución de tiempo en la detección de cualquier anormalidad permitiendo salvar muchas vidas de ganado y en ese orden de ideas evitando pérdidas tanto en la industria como en la organización.

# PROBLEMA

El problema en el que estamos situados consiste en comprimir las imágenes para clasificar la salud de los animales en el contexto de la ganadería de precisión, con esto se busca crear un método en el cual se realice una detección temprana de enfermedades en el ganado, por consiguiente, será útil para implementar un plan de acción y atender de manera oportuna al ganado que así lo requiera. Previniendo consecuencias como la muerte temprana de estos animales y las pérdidas que esto conlleva a los ganaderos.

# RESUMEN:

el objetivo de este informe es analizar y solucionar el problema de la compresión y descomprensión eficiente de datos, más específicamente de imágenes. Lo que se pretende es el diseño y creación de un algoritmo, que haciendo uso de tablas hash, árboles o de ambos; nos permita comprimir y descomprimir imágenes teniendo en cuenta que se debe lograr la mayor tasa de compresión sin descuidar la exactitud de la clasificación, además dicho algoritmo debe optimizar el consumo de tiempo y de memoria. Teniendo como prioridad en los dos últimos al tiempo. Esto con la finalidad de reducir el consumo de energía.

Con la inclusión de la tecnología al campo se abren un sinfín de herramientas para optimizar la producción de alimentos, pero esto conlleva a nuevos desafíos, por ejemplo, muchos de los datos de los ganaderos se encuentran almacenados de forma física en cuadernos o en sistemas de información zootecnistas. realizar la digitalización y clasificación de los datos útiles para la ganadería de precisión es todo un desafío, además se debe garantizar que dicho sistema de almacenamiento sea eficiente en términos de consumo de energía, para lograrlo los ganaderos tienen que fusionar los datos de diferentes fuentes, reducir la dimensión de los datos, minimizar el consumo de energía en los equipos de la granja, y comprimir los datos, aquí es donde los algoritmos de compresión de datos hacen su entrada ya que al disminuir los tiempos de cómputo, el consumo de energía se ve disminuido.

La resolución de este problema es de gran importancia, uno de los problemas más grandes de la ganadería es la gran huella de carbono que esta produce, este problema es mitigado gracias a la ganadería de precisión, ya que logra optimizar la producción y disminuir costos, al usar algoritmos de compresión obtenemos datos menos pesados, lo que facilita su transmisión por el ancho de banda del internet, lo que permite el almacenamiento en la nube y permite difundir los datos para su estudio. Todo esto se resume en nuevas herramientas y una reducción de costos para el ganadero.

# PROBLEMAS RELACIONADOS

# UNA PLATAFORMA DE BIENESTAR ANIMAL PARA LA GANADERÍA EXTENSIVA

Se presenta un análisis en relación a la ganadería y al cuidado animal, en base a un estudio que plantea que los animales de granja deberían tener mejores condiciones de vida, pues se pone como evidencia que hace un siglo la cobertura para la producción agrícola solo cubría a las necesidades del agricultor mas no a los animales y en las últimas décadas, se generó una preocupación a escala mundial de parte de los consumidores, que aseguran que los animales deben estar en aptas condiciones para generar mejores productos.

Con favor de los recientes avances tecnológicos, se plantea como solución un dispositivo que pueda cubrir la actividad de los animales, ofreciendo indicadores de su salud, estado de ánimo, comportamiento, presión arterial, movimiento, dieta, etc.

Este dispositivo se considera venir en forma de collar para la comodidad del animal, contara con un sensor capaz de ofrecer al usuario, es decir, al agricultor, la información basta del animal, desde su salud hasta su ubicación en tiempo real. El aparato anteriormente mencionado, traería un sistema que almacenaría información en todo momento, para tener al tanto al usuario, de los comportamientos de los animales, desde cambios en su entorno al bienestar del grupo de los animales, para en caso de haber una alteración en cualquier aspecto ya mencionado, tener la mejor y pronta intervención posible.

Imagen que contiene interior, tabla, cuarto

Descripción generada automáticamente

Algoritmo:

Los consumidores desean mejores condiciones de vida para los animales granjeros, para así tener mejor calidad de productos, se plantea la solución de un dispositivo que pueda asimilar todo aspecto de los animales, desde tema de salud, asimilación de comportamientos y alerta de alguna incongruencia para una temprana intervención, en la forma de collar para la comodidad del animal, para evitar altos costos, se implementan procesadores que puedan subir y compilar información desde corta a larga distancia y con una batería recargable pero el sustento del usuario.

Diagrama

Descripción generada automáticamente

Componentes:

Habiendo considerado todas las limitaciones mencionadas anteriormente, nos decidimos por la unidad de medición inercial BMI160 de Bosch. El orquestador seleccionado entre todos los componentes es el microcontrolador de baja potencia STM32L162 ARM-Cortex-M3, con el plan de asumir, en el futuro, parte del estrés computacional del Edge Deviee, o incluso posiblemente ejecutar un Artificial entrenado. Todos los componentes fueron seleccionados para cumplir con los requisitos de potencia establecidos, así como las restricciones de dimensión.

# VISUAL LOCALISATION AND INDIVIDUAL IDENTIFICATION OF HOLSTEIN FRIESIAN CATTLE VIA DEEP LEARNING:

El Ganado de Raza Holstein es ampliamente criado en la ganadería gracias a que es la raza con la mayor producción de leche del tipo bovina, esto le ha dado una gran importancia económica, uno de los factores que define la productividad de la raza es su genética, por eso a la hora de exportación uno de los requisitos es la identificación y trazabilidad del animal para así confirmar parte de su genética, por lo general esta identificación se realiza con tatuajes, chips dentro del animal u orejeras entre muchos otros. Pero estos métodos han generado preocupación respecto al bienestar del animal, por ejemplo, las orejeras pueden causar danos físicos al animal. Gracias a estas razones esta investigación busca automatizar el reconocimiento del ganado por medio de los patrones que este presenta en su pelaje, con el objetivo de tener una granja más eficaz y beneficiando al bienestar del animal.

Primero se expusieron algunos métodos para la identificación biométrica de ganado bovino como por ejemplo la construcción de modelos faciales del animal implementado un mapeo del rostro del animal, el escaneo de la reina o del cuerpo del animal. Otro ejemplo es la identificación del animal por medio de los patrones del hocico del animal (Cattle muzzle patterns). Por último, se expone el uso de escaneos de los patrones del pelaje del ganado bovino para lograr esto se implementaron los algoritmos Speeded-Up Robust features (SURF) y el Scale Invariant Features (SIFT).

Para resolver esta problemática se implemento el algoritmo Regional Convolutional Neural Networks (R-CNN) este algoritmo detecta al animal y determina su posición luego genera un id buscando los patrones del pelaje que hay en el dorso del animal. Posteriormente toda esta información es guardada en forma secuencia gracias a las redes LSTM (Long Short Term Memory), esta red se encarga de organizar la información.

Tabla

Descripción generada automáticamente

Podemos ver que la precisión promedio es de 99,3%.

# GANADERÍA DE PRECISIÓN EN VACUNO DE CARNE:

La universidad de zaragoza realiza una revisión de varias herramientas que se usan en la ganadería de precisión para la producción de carne, luego de la selección se expone el funcionamiento de dichos resultados, varias herramientas se enfocan en la identificación del animal haciendo uso de varios métodos como la identificación con radiofrecuencia (RFID) y la identificación a través de parámetros biométricos o morfológicos con el análisis de imágenes. Es importante resaltar la importancia de la identificación del animal en la ganadería de precisión ya que dicho seguimiento le brinda al ganadero herramientas para obtener información rápida y precisa del animal lo que permite facilitar la trazabilidad de del producto que llega al consumidor, implementar vigilancia, mejorar prácticas de gestión productiva, entre muchos otros beneficios.

La identificación por radiofrecuencia realiza una captura automática de datos e identifica animales, objetos y personas. Para lograrlo se hace uso de dispositivos llamados etiquetas (tags). Estas etiquetas pueden ser placas de identificación para las orejas, chips subcutáneos o bolos de ingestión animal. Dichas etiquetas poseen un numero único o ID que permite acceder a la información de un animal es especifico, pero, se observan dificultades al emplear esta herramienta en ganaderos con uno o dos animales. Además, se debe tener en cuenta varios factores como el tipo de granja, la infraestructura de esta, la cantidad y la especie del animal.

La identificación por parámetros biométricos y morfológicos hace uso de algoritmos para el análisis de imágenes en el caso de los bovinos se procesan impresiones del hocico para la identificación del animal, otros modelos trabajan con imágenes faciales del animal e incluso existe un modelo que hace un reconocimiento del iris del animal, todos estos métodos no lastiman al animal.

* 1. **TABLAS HASH**

Una tabla hash, matriz asociativa, hashing, mapa hash, tabla de dispersión o tabla fragmentada es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos almacenados a partir de una clave generada . Las tablas hash se suelen implementar sobre vectores de una dimensión, aunque se pueden hacer implementaciones multi-dimensionales basadas en varias claves. Como en el caso de los arrays, las tablas hash proveen tiempo constante de búsqueda promedio O,1​ sin importar el número de elementos en la tabla. Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información.

Las tablas hash almacenan la información en posiciones pseudo-aleatorias, así que el acceso ordenado a su contenido es bastante lento.

Ejemplo de aplicación:

Un ejemplo práctico para ilustrar qué es una tabla hash es el siguiente: Se necesita organizar los periódicos que llegan diariamente de tal forma que se puedan ubicar de forma rápida, entonces se hace lo siguiente: se hace una gran caja para guardar todos los periódicos (una tabla), y se divide en 31 contenedores (ahora es un hash table o tabla fragmentada), y la clave para guardar los periódicos es el día de publicación (índice). Cuando se requiere buscar un periódico se busca por el día que fue publicado y así se sabe en qué zócalo (bucket) está. Varios periódicos quedarán guardados en el mismo zócalo (es decir, colisionan al ser almacenados), lo que implica buscar en la sub-lista que se guarda en cada zócalo. De esta forma se reduce el tamaño de las búsquedas de O(n) a O(log(n)).

Inserción

La forma de implementar en función esta operación es pidiendo la llave y el valor, para con estos poder hacer la inserción del dato. Esto se consigue aplicando la función resumen (hash) a la clave del elemento. El resultado de la función resumen ha de mapearse al espacio de direcciones del vector que se emplea como soporte, lo cual se consigue con la función módulo. Este problema se puede solucionar asociando una lista a cada posición de la tabla, aplicando otra función o buscando el siguiente elemento libre.

Interfaz de usuario gráfica

Descripción generada automáticamente

Búsqueda

Para recuperar los datos, es necesario únicamente conocer la clave del elemento, a la cual se le aplica la función resumen. Si el elemento existente en la posición indicada en el paso anterior tiene la misma clave que la empleada en la búsqueda, entonces es el deseado. Si la clave es distinta, se ha de buscar el elemento según la técnica empleada para resolver el problema de las colisiones al almacenar el elemento. Algunas tablas hash permiten almacenar múltiples valores bajo la misma clave.

Prácticas recomendadas para las funciones hash

Una buena función hash es esencial para el buen rendimiento de una tabla hash. Las colisiones son generalmente resueltas por algún tipo de búsqueda lineal, así que, si la función tiende a generar valores similares, las búsquedas resultantes se vuelven lentas.

En una función hash ideal, el cambio de un simple bit en la llave debería cambiar la mitad de los bits del hash, y este cambio debería ser independiente de los cambios provocados por otros bits de la llave. Es deseable utilizar la misma función hash para arrays de cualquier tamaño concebible.

Un valor hash genérico es calculado, llenando un entero natural de máquina.

Este valor es reducido a un índice válido en el vector encontrando su módulo con respecto al tamaño del array.

Sin embargo, el uso de una tabla de tamaño primo no es un sustituto a una buena función hash.

Un problema bastante común que ocurre con las funciones hash es el aglomeramiento.

Para usar una tabla hash se necesita:

* Una estructura de acceso directo (normalmente un array).
* Una estructura de datos con una clave
* Una función resumen (hash) cuyo dominio sea el espacio de claves y su imagen (o rango) los números naturales.

# ALTERNATIVAS DE SOLUCION

**ALGORITMOS CON PERDIDA**

Se denomina algoritmo de compresión sin pérdida a cualquier procedimiento de codificación que tenga como objetivo representar cierta cantidad de información utilizando u ocupando un espacio menor, siendo posible una reconstrucción exacta de los datos originales.

# ALGORITMO DE HUFFMAN

El algoritmo de huffman fue desarrollado por David A Huffman en 1952 e hizo su aparición en “A *Method for the Construction of Minimum-Redundancy Codes*.”. Este algoritmo de compresión está diseñado para minimizar el promedio de bits necesario para transmitir un carácter, esto cuando se quieren varias copias independientes y estadísticamente equivalentes de dicho carácter. Este algoritmo nos proporciona una representación para cada carácter, con la particularidad que la cadena de bits que representa a este carácter nunca es prefija de la cadena de bits de otro carácter distinto (código prefijo). Para lograr esto el algoritmo hace uso de un árbol binario que tiene cada uno de los caracteres como sus hojas y se forma de manera que si lo seguimos por la raíz a cada una de sus hojas obtendremos el código en binario asociado a él.

Para poder construir el árbol debemos ordenar a los caracteres únicos por su frecuencia relativa, más específicamente los ordenamos de izquierda a derecha, donde los que estén a la izquierda tienen menor frecuencia, luego cada carácter tendrá junto con su frecuencia un nodo conectado por ramas. Luego formamos un nodo intermedio sumando a los 2 nodos que estén a la izquierda, es decir los nodos con menor peso, el nuevo nodo tendrá como peso la suma de las frecuencias de los nodos con menor peso además tendrá a estos nodos como hijos y en la lista este nodo intermedio reemplazara a los nodos hijos, repetiremos este proceso varias veces hasta solo tener un nodo en la lista y dicho nodo será la raíz del árbol de Hoffman. Por último, etiquetamos todas las ramas que conectan los nodos, las ramas con pendientes positivas las vamos a etiquetar con 0 y las ramas con pendientes negativas están etiquetadas con 1.

Ahora podemos leer la nueva representación de bits para cada carácter del árbol de Huffman solo debemos recorrer las ramas hasta llegar al carácter deseado y vamos asignando unos o ceros según nos lo indique las etiquetas del camino que tomemos.

Veamos un ejemplo:

Codificación de huffman para almacenar la palabra “abracadabra” paso por paso

1. Ordenamos los caracteres únicos por su frecuencia relativa, y los ponemos dentro de nodos:

Diagrama

Descripción generada automáticamente

1. Ahora tomamos los 2 nodos con menor frecuencia relativa y sumamos sus valores para obtener crear un nuevo nodo, recordemos que este nodo debe tener como hijos a los nodos d y c. luego agregamos este nodo al árbol:

Diagrama

Descripción generada automáticamente

1. Repetimos el lo hecho en el segundo paso, es decir sumamos a los dos nodos de menor frecuencia y creamos un nuevo nodo que tiene 2 hijos. Luego ordenamos los nodos del nivel superior según su frecuencia relativa de menor a mayor:

Diagrama

Descripción generada automáticamente

1. Repetimos los pasos uno 2 y 3 hasta llegar a nuestro nodo raíz con frecuencia relativa 1, luego marcamos las ramas del árbol con unos y ceros:

Diagrama

Descripción generada automáticamente

1. Luego seguimos la ruta hasta llegar a cada uno de nuestros nodos y la representación será la recopilación de las etiquetas que recorramos hasta llegar a nuestro carácter deseado por ejemplo la representación de “a” será “0”, podemos ver la diferencia entre la representación binaria y la representación con el algoritmo de huffman:

Tabla

Descripción generada automáticamente

Con el código de Huffman podemos escribir la palabra abracadabra mediante la siguiente cadena de 0 y 1:

0 11 101 0 1000 0 1001 0 11 101 0

a b r a d a d a b r a

# TRANSFORMADA DE BURRORWS-WHEELER:

La transformada Burrorws-Wheeler (abreviado BWT) es un algoritmo de compresión usado generalmente en la web y en Linux, este algoritmo toma un bloque de datos y lo ordena de forma que el bloque de salida tiene exactamente los mismos elementos de datos con los que empezó, solo que su orden cambia, esta transformación es reversible, es decir podemos volver al bloque de datos original sin pérdida de fidelidad, este algoritmo ordena, en su gran mayoría, a los datos o caracteres repetidos en grupos para que así sean más comprensibles para otros algoritmos, a veces se le considera como un catalizador para la compresión, veamos un ejemplo:

Primero vamos a definir la rotación de un String lo que hacemos es que dado un String vamos a mover sus caracteres una posición a la izquierda y su primer carácter ocupara el final de la lista por ejemplo si tengo la palabra “perla” su primera rotación seria “erlap”, para aplicar la BWT debemos las n rotaciones de nuestro String (donde n es la longitud del String) y siempre nuestra primera rotación es el Sting original.

Ej:

“perla”

“erlap”

“rlape”

“laper”

“aperl”

Para poder aplicar la transformada primero debemos definir un símbolo “terminador” este símbolo debe ser diferente a todos los símbolos presentes en la cadena. Luego definimos que este símbolo es menor alfabéticamente respecto a los símbolos de nuestra cadena.

Este símbolo lo concatenaremos al final de nuestra cadena, ya que nos ayuda a evitar algunos problemas por ejemplo trate de obtener las rotaciones del String “bonbon” nos encontraremos con que algunas rotaciones son iguales, pero al agregar un símbolo terminador (usaremos “$” en este ejemplo) garantizamos que no obtendremos rotaciones 2 rotaciones iguales:

“bonbon” “bonbon$”

“onbonb” “$bonbon”

“nbonbo” “n$bonbo”

“bonbon” “on$bonb”

“onbonb” “’bon$bon”

“nbonbo” “nbon$bo”

“onbon$b”

Así pues el primer paso de la transformada será obtener todas las rotaciones del string y luego vamos a organizar dichas rotaciones según el orden alfabético:

Diagrama

Descripción generada automáticamente con confianza baja  
  
de esta forma obtendremos la transformada de Burrows-Wheeler, esta será la última columna de la matriz, como podemos ver que este String es una permutación del String original.

Ahora si queremos volver a nuestro String original, debemos hacer lo siguiente:

Tomamos la última columna de la matriz BWT, esta columna tiene todos los caracteres del String así que los ordenamos alfabéticamente para si obtener la primera columna, luego volvemos a agregar la última columna de la matriz, teniendo en cuenta el primer par de caracteres organizamos de forma alfabética, y repetimos este proceso hasta reconstruir la matriz BWT y la fila que tenga el carácter terminador al final será el texto original

# CODIFICACIÓN DE LONGITUD DE EJECUCIÓN (RLE)

La codificación de longitud de ejecución (RLE) es una forma muy simple de compresión de datos en la que se proporciona un flujo de datos como entrada (es decir, "AAABBCCCC") y la salida es una secuencia de conteos de valores de datos consecutivos en una fila (es decir, " 3A2B4C ").

La codificación de longitud de ejecución es una forma muy simple de compresión de datos en la que se proporciona un flujo de datos como entrada y la salida es una secuencia de conteos de valores de datos consecutivos en una fila . Su simplicidad tanto en la codificación como en la decodificación es una de las características más atractivas del algoritmo.

Como habrá notado, mientras más valores consecutivos haya en una fila, más espacio ahorraremos en la compresión resultante. De hecho, incluso podríamos aumentar el tamaño de nuestros datos, ya que una sola instancia de un carácter da como resultado 2 caracteres en la salida de la codificación.  
Por ejemplo, la cámara Pixy , que es una cámara robótica que le ayuda a rastrear fácilmente los objetos, usa RLE para comprimir los datos de video etiquetados antes de transferirlos desde el dispositivo de cámara integrado a una aplicación externa.

**Codificación**

Para codificar una cadena de datos, el código tendrá que recorrer cada carácter de los datos y contar las ocurrencias.

A continuación, encontrará una implementación simple en Python:

Texto

Descripción generada automáticamente

A partir de los comentarios, debe poder decir lo que está pasando a lo largo del código. Si no es así, sería un buen ejercicio ejecutar el código con un depurador y verlo en acción.

Continuando con el mismo archivo anterior, aquí hay un ejemplo del código que se está ejecutando:

encoded\_val = rle\_encode('AAAAAAFDDCCCCCCCAEEEEEEEEEEEEEEEEE') print(encoded\_val)

Y la salida:

$ python rle-encode.py 6A1F2D7C1A17E

Decodificación

Si ve un carácter numérico, luego lo agrega a su cont, y si ve un carácter no numérico luego lo agrega a su cont, entonces agrega esos caracteres a su decodificación, que se devuelve a la persona que llama una vez que se repite en todas las entradas data.

**Texto

Descripción generada automáticamente**

Entrada

decoded\_val = rle\_decode('6A1F2D7C1A17E')

print(decoded\_val)

Salida

$ python rle-decode.py

AAAAAAFDDCCCCCCCAEEEEEEEEEEEEEEEEE

# CODIFICACIÓN ARITMÉTICA

La codificación aritmética es una forma de codificación entrópica utilizado en compresión sin pérdidas. Normalmente, una cadena de caracteres como las palabras «hola allí» está representada utilizando un número fijo de bits por carácter, como en el código ASCII. Cuando una cadena es convertida a codificación aritmética, los caracteres frecuentemente usados serán almacenados con menos bits y los no-tan-frecuentemente utilizados caracteres serán almacenados con más bits, resultando en menos bits utilizados en total.

En el caso más simple, la probabilidad de aparición de cada símbolo es igual. Una solución más eficiente es representar una secuencia de estos tres símbolos como un número racional en base 3 donde cada dígito representa un símbolo. Esto es factible para secuencias largas porque hay algoritmos eficientes para convertir la base de números precisos arbitrariamente. Para decodificar el valor, conociendo que la cadena original tenía longitud 6, uno puede simplemente convertir de vuelta a base 3, redondear a 6 dígitos y recuperar la cadena.

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.

60% de probabilidad del símbolo NEUTRAL

20% de probabilidad del símbolo POSITIVE

10% de probabilidad del símbolo NEGATIVE

10% de probabilidad del símbolo END-OF-DATA (Fin de los datos).

Los modelos también pueden manejar alfabetos diferentes al simple conjunto de cuatro símbolos escogido para este ejemplo. Los modelos pueden ser incluso adaptativos, de forma que continuamente cambian su predicción de los datos basados en lo que el flujo de datos contiene actualmente. El decodificador debe tener el mismo modelo que el codificador.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Codificación y decodificación: perspectiva general

El siguiente símbolo que necesita ser codificado.

El intervalo actual (al inicio del proceso de codificación, el intervalo es [0,1], pero eso cambiará).

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.

**Ejemplo**: para el modelo de cuatro símbolos de arriba:

* el intervalo para NEUTRAL sería [0, 0.6)
* el intervalo para POSITIVE sería [0.6, 0.8)
* el intervalo para NEGATIVE sería [0.8, 0.9)
* el intervalo para END-OF-DATA sería [0.9, 1).

Cuando todos los símbolos hayan sido codificados, el intervalo resultante inequívocamente identifica la secuencia de símbolos que lo produjeron. Cualquiera que tenga el mismo intervalo final y el modelo que es usado puede reconstruir la secuencia de símbolos que debe haber sido ingresada al codificador para resultar en ese intervalo final.

No es necesario transmitir el intervalo final, sin embargo, es sólo necesario transmitir una fracción que cae dentro del intervalo. En particular, sólo es necesario transmitir suficientes dígitos (en cualquier base) de la fracción de forma que todas las fracciones que comienzan con esos dígitos caigan dentro del intervalo final; esto garantizará que el código resultante es un código prefijo.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

# ALGORITMOS CON PERDIDA

Algoritmo de compresión con pérdida se refiere a cualquier procedimiento de codificación que tenga como objetivo representar cierta cantidad de información utilizando una menor cantidad de la misma, siendo imposible una reconstrucción exacta de los datos originales.

# LA TRANSFORMADA DISCRETA DEL COSENO (DCT)

Cuenta con una buena propiedad de compactación de energía y es muy similar a la KLT Karhunen-LoèveTransform), que produce coeficientes incorrelados, con la diferencia de que los vectores base de la DCT dependen sólo del orden de la transformada seleccionado, y no de las propiedades estadísticas de los datos de entrada.

La decorrelación de coeficientes es muy importante para compresión, ya que, el posterior tratamiento de cada coeficiente se puede realizar de forma independiente, sin pérdida de eficiencia de compresión. Otro aspecto importante de la DCT es la capacidad de cuantificar los coeficientes utilizando valores de cuantificación que se eligen de forma visual.

La DCT está bastante relacionada con la DFT, con la diferencia de que es una transformada real, debido a que los vectores base se componen exclusivamente de funciones coseno muestreadas.

DCT UNIDIMENSIONAL

La respuesta del sistema visual humano depende de la frecuencia espacial. Si pudiéramos, de algún modo descomponer una imagen en un conjunto de imágenes, cada una con una frecuencia espacial particular, podríamos separar la estructura de la imagen que el ojo puede ver a partir de la estructura que es imperceptible. La DCT puede proporcionar una buena aproximación a esta descomposición.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

2.73 representa un conjunto de ocho funciones base co/sinusoidales de amplitud uniforme, cada una muestreada en ocho puntos. La forma de onda superior izquierda es simplemente una constante, mientras que las otras siete presentan un comportamiento alterno a frecuencias más altas progresivamente. El objetivo es descomponer estas ocho muestras en un conjunto de formas de onda de diferentes frecuencias espaciales. Los coeficientes C son representados gráficamente en la Fig.

2.74 presenta una secuencia en la que las ocho formas de onda ponderadas son sumadas progresivamente, comenzando con la frecuencia más baja, hasta que finalmente se reconstruye el conjunto original de muestras.

Los coeficientes representados en la Fig. 2.72 son la salida de una DCT de 8-puntos para valores de ocho muestras en la Fig. El coeficiente que pondera la función base constante se denomina coeficiente DC.

El resto de los coeficientes se denominan coeficientes AC. Observar que el término DC proporciona el promedio sobre el conjunto de muestras

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Las formas de onda ortogonales son independientes. no existe la posibilidad de que una forma de onda dada, pueda representarse a través de una combinación de las demás formas de onda.

# JEPG

El algoritmo JPEG es un algoritmo de compresión usado para imágenes. Este algoritmo hace parte de los algoritmos de compresión con perdida, es decir que pierde información, y por consiguiente calidad de imagen, pero este se aprovecha de los defectos del ojo humano para que no podamos notar la diferencia, este algoritmo puede lograr una compresión del archivo original de hasta 10 veces sin que el ojo humano note mucho la diferencia.

Este algoritmo empieza tomando cada píxel de la imagen original y haciendo uso del modelo de color RGB (en la mayoría de los casos) vuelve a cada color del pixel en números, el RGB le da 3 valores al color, estos tres valores son uno de rojo, verde y uno de azul y cada uno va de 0 a 255. Así convierte el color de cada píxel a 3 números para poder interpretarlo.

Luego divide a la imagen en trozos de 8 pixeles por 8 pixeles y de aquí en adelante va a trabajar con cada uno de estos trozos y además actúa con los 64 valores individuales de cada color. Por ejemplo, JPEG tiene una matriz o imagen con 64 valores de rojo, es decir con 64 intensidades de rojo a esta imagen se le aplica la transformada discreta del coseno en 2 dimensiones y cuantificación perceptual.

Recordemos que gracias a las ondas coseno podemos combinar varias ondas coseno para obtener otra onda más complicada, esto también se aplica en ondas de 2 dimisiones ahora si tomamos estas ondas de dos dimensiones y las pintamos como una imagen, luego las miramos desde arriba y pintamos la onda donde lo más claro esta más arriba y lo más oscuro está más abajo obtenemos esto:

Asi podemos ver que nuestra imagen del color rojo es parecida a una onda de 2 dimisiones, lo que nos lleva a que podemos representar a nuestra imagen haciendo uso de otras ondas coseno más sencillas, en JPEG se hace uso de 64 ondas coseno. Entonces JPEG calcula los coeficientes de cada una de las ondas para obtener nuestra imagen, así JPEG divide los valores de la matriz de rojo por los valores de otra tabla, la tabla de cuantización, así estos resultados se redondean y se almacenan.

Visualmente lo que le hace a la imagen es suavizar las variaciones bruscas de brillo y de color. Es como aplicar una especie de desenfoque imperceptible a la imagen. Esto se hace así debido a que la visión humana es menos sensible a las grandes variaciones en una zona pequeña que a las pequeñas variaciones en una zona amplia. Esta eliminación selectiva de información se hace en base a estudios estadísticos de la visión humana, realizados a base de encuestar a personas. Por último se usa el algoritmo de huffman para comprimir la matriz.

# MP3

Para comprender el funcionamiento del algoritmo Mp3 primero debemos explicar el concepto de enmascaramiento sonoro, a grandes rasgos el enmascaramiento sonoro sucede cuando por ejemplo vamos por la calle conversando con alguien y de repente pasa un auto a gran velocidad, en ese momento dejamos de oír a nuestro acompañante, lo que sucede es que nuestro oído le da prioridad al sonido producido por el auto e ignora el sonido de la conversación de nuestro acompañante. Siendo más precisos un sonido puede enmascarar a otro cuando llegan simultáneamente al oído esto depende de sus frecuencias y de los volúmenes relativos.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Además, existe un enmascaramiento temporal esto sucede cuando un sonido tiene suficiente potencia para enmascarar a otro hay unos instantes antes y después en los que el oído humano no percibe otros sonidos, esto depende de cuan cerca estén en el tiempo y de su volumen relativo.

Gráfico, Diagrama

Descripción generada automáticamente con confianza media

Ahora bien lo que hace el algoritmo es dividir el audio en muchos fragmentos de corta duración, luego en cada uno de estos fragmentos de busca si tienen algún sonido enmascararte que este “ocultando” a otros sonidos de las otras bandas del fragmento, si es así dicho fragmento es codificado con menos bits ya que dichos sonidos no van a ser percibidos por nuestro oído, aunque la cantidad de bits también dependerá de la calidad que se busque en la compresión pero por lo general si se busca comprimir con alta calidad se reducirá la resolución del fragmento lo justo para que el sonido de fondo sea “ocultado” por el sonido enmascarante.

Al terminar nuestro fragmento será codificado ocupando menos información que el original, así que el proceso se repetirá con todos los fragmentos en los que el algoritmo dividió al archivo original para comprimir a todo el audio y se “reconstruye” al audio en un archivo comprimido.

Al final se le aplica una codificación de huffman.

# 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.

Una representación de imagen fractal puede ser descrita matemáticamente como un [sistema de funciones iteradas](https://es.wikipedia.org/wiki/Sistema_de_funciones_iteradas) (SFI).

**Para imágenes binarias**

Comenzando con la representación de una imagen binaria, donde la imagen puede ser descrita como un subconjunto de {\displaystyle \mathbb {R} ^{2}}R^2. Un SFI es un conjunto de [contracciones](https://es.wikipedia.org/wiki/Contracci%C3%B3n_(espacio_m%C3%A9trico)) *ƒ*1,...,*ƒN*,

{\displaystyle f\_{i}:\mathbb {R} ^{2}\to \mathbb {R} ^{2}.}

De acuerdo con estas funciones, el SFI describe un conjunto bidimensional *S* como el punto fijo del [operador de Hutchinson](https://es.wikipedia.org/wiki/Operador_de_Hutchinson)

{\displaystyle H(A)=\bigcup \_{i=1}^{N}f\_{i}(A),\quad A\subset \mathbb {R} ^{2}.}Texto, Carta

Descripción generada automáticamente

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](https://es.wikipedia.org/wiki/M%C3%A9todo_del_punto_fijo): para cualquier conjunto inicial [compacto](https://es.wikipedia.org/wiki/Espacio_compacto) no vacío *A*0, 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:

{\displaystyle S=f\_{1}(S)\cup f\_{2}(S)\cup \cdots \cup f\_{N}(S)}

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

**Extensión a la escala de grises**

La representación SFI puede ser extendida a una imagen en [escala de grises](https://es.wikipedia.org/wiki/Escala_de_grises) considerando la [gráfica](https://es.wikipedia.org/wiki/Gr%C3%A1fica_de_una_funci%C3%B3n) de la imagen como un subconjunto de {\displaystyle \mathbb {R} ^{3}}. 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 *ƒ*1,...,*ƒN*, pero en R^3



{\displaystyle f\_{i}:\mathbb {R} ^{3}\to \mathbb {R} ^{3}.}

**Codificación**

Un problema de la investigación sobre la representación fractal de la imagen es como elegir *ƒ*1,...,*ƒN* de tal forma que su punto fijo se aproxime a la imagen de entrada, y cómo hacer esto de forma eficiente. Una simple aproximación[1](https://es.wikipedia.org/wiki/Compresi%C3%B3n_fractal#cite_note-1)​ para hacer esto es la siguiente:

1. Particionar la imagen en bloques *Ri* de tamaño *s*×*s*.
2. Para cada *Ri*, buscar en la imagen para encontrar un bloque *Di* de tamaño 2*s*×2*s* que sea similar a *Ri*.
3. Seleccionar las funciones de mapeo tal que *H*(*Di*) = *Ri* para cada *i*.

En el segundo paso, es importante encontrar un bloque similar para que el SFI represente fielmente la imagen de entrada, así que necesitan ser considerados un número suficiente de bloques candidatos para *Di*. Por otro lado, una larga búsqueda considerando muchos bloques es computacionalmente costoso.

{\displaystyle f\_{i}:\mathbb {R} ^{2}\to \mathbb {R} ^{2}.}

**características**

Con la compresión fractal, la codificación es extremadamente cara a nivel computacional debido a la búsqueda de similitudes propias. 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.2​3​

4​ A ratios de compresión elevados, la compresión fractal puede ofrecer una calidad superior. Para imágenes por satélite, se han utilizado ratios 170:1 con resultados aceptables.5​ La compresión fractal de vídeo en ratios de 25:1 a 244:1 se han realizado en tiempos razonables .6

# REFERENCIAS:

[Ber14] Daniel Berckmans. Precision livestock farming technologies for welfare management in intensive livestock systems. Revue scientifique et technique (International Office of Epizootics), 33:189–196, April 2014.

[CLF+19] Min Chen, Wei Li, Giancarlo Fortino, Yixue Hao, Long Hu, and Iztok Humar. A dynamic service migration mechanism in edge cognitive computing. ACM Transactions on Internet Technology, 19(2):30:1–30:15, April 2019.

[Fed18] Federation of Veterinarians of Europe. Monitoring of farm animal welfare using animal indicators, November 2018.

[Foo18] Food and Agriculture Organization, Animal Production and Health Division (AGA). Shaping the future of livestock, January 2018.

[IV15] A. Ilapakurti and C. Vuppalapati. Building an iot framework for connected dairy. In Proc. IEEE First Int. Conf. Big Data Computing Service and Applications, pages 275–285. IEEE, March 2015.

[KLH+19] Y. Kaurivi, Richard Laven, Rebecca Hickson, Kevin Stafford, and Tim Parkinson. Identification of suitable animal welfare assessment measures for extensive beef systems in new zealand. Agriculture, 9(3):66, March 2019.

[MCL+17] Ariel Marcel Tarazona Morales, Maria Camila Ceballos, Guillermo Correa Londoo, Csar Augusto Cuartas Cardona, Juan Fernando Naranjo Ramrez, and Mateus Jos Rodrigues Paranhos da Costa. Welfare of cattle kept in intensive silvopastoral systems: A case report. Revista Brasileira de Zootecnia, 46(6):478–488, June 2017.

[MW14] D. J. Mellor and J. R. Webster. Development of animal welfare understanding drives change in mínimum welfare standards. Revue scientifique et technique (International Office of Epizootics), 33:121–130, April 2014.

[NTCG18] Luis N´obrega, Andre Tavares, Antonio Cardoso, and Pedro Gon¸calves. Animal monitoring based on IoT technologies. In 2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany), pages 1–5. IEEE, May 2018.

[wM17] Ciira wa Maina. IoT at the grassroots — exploring the use of sensors for livestock monitoring. In 2017 IST-Africa Week Conference (IST-Africa), pages 1–8. IEEE, May 2017.

[WYC+18] Y. Wang, X. Yong, Z. Chen, H. Zheng, J. Zhuang, and J. Liu. The design of an intelligent livestock production monitoring and management system. In 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), pages 944–948. IEEE, May 2018.

Fischer, Y. (1992). «Fractal Image Compression». SIGGRAPH'92 course notes.

Alternativas a la compresión de imagen y consideraciones sobre almacenamiento de medios Archivado el 3 de marzo de 2008 en Wayback Machine. Georgia Institute of Technology reference to compression/decompression time.

Heath, Steve (1999). Multimedia and Communications Technology. Focal Press. pp. 120-123. ISBN 0240515293.

Sayood, Khalid (2005). Introducción a la Compresión de Datos, Tercera Edición. Morgan Kaufmann. pp. 560-569. ISBN 012620862X.

Geoscience and Remote Sensing Symposium paper Achieving high data compression of self-similar satellite images using fractal.

Codificación fractal de secuencias de vídeo

Veenadevi. SV, AG Ananth, "Compresión de imágenes fractal de imágenes de satélite", Journal of Computer Applications, septiembre de 2011.

[2] Hitashi Gaganpreet Kaur Sugandha Sharma, "Compresión de imágenes fractal: una revisión", Revista internacional de investigación avanzada en informática y software Engineering, febrero de 2014.

[3] Xinpeng Zhang, Yanli Ren, Liquan Shen, Zhenxing Qian y Guorui Feng, "Compressing Encrypted Images With Auxiliary Information", Multimedia, agosto de 2014.

[4] Dr. SS Pandey, Manu Pratap Singh, Vikas Pandey, "Block Wise Image Compression & Reduced Blocks Artifacts Using Discrete International Journal Publications, marzo de 2015. Transacciones internacionales IEEE sobre la transformación coseno de Bruylants científicos y de investigación, AdrianMunteanu," Compresión volumétrica de imágenes médicas basada en wavelet ",

[5] Tim PeterSchelkens, Elsevier, diciembre de 2014.

[6] Navnidhi Chaturvedi, Dr. SJ Basha, "Comparación de métodos de marca de agua de imagen digital DWT y DWT- DCT sobre la base de PSNR", International Jo urnal of Innovative Research in Science, Engineering and Technology, diciembre de 2012. 'SL

[7] Nigar Sulthana, Mahesh Chandra, "Compresión de imágenes con codificación aritmética adaptativa", Revista internacional de aplicaciones informáticas, 2011.

[8] Rafeeq AL-HASHEMI, Ayman AL -DMOUR, Fares FRAIJ, Ahmed MUSA, "Una técnica de compresión de imágenes semi-sin pérdida en escala de grises usando RLE", Revista de Ciencias de la Computación Aplicadas y Matemáticas, 2011.

[9] Sharan Kumar, Dr. Jayadevappa, Santosh .D.Bhopale, Radhika. Naik, "Implementación de Huffman Image Compression Revista internacional de investigación innovadora en ingeniería eléctrica, electrónica, instrumentación y control, abril de 2014. Algoritmo de descompresión puy",