# **TÍTULO (UNA BREVE DESCRIPCIÓN DEL PROYECTO, ENTRE 8 Y 12 PALABRAS)**

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre del primer autor  Universidad ...  País  Correo electrónico en Eafit | Nombre del segundo autor  Universidad ...  País  Correo electrónico en Eafit | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

**Para cada versión de este informe: 1. Elimine todo el texto en rojo. 2. Ajustar los espacios entre las palabras y los párrafos. 3. Cambiar el color de todos los textos a negro.**

**Texto rojo =** Comentarios

**Texto negro =** Ccontribución de Simón y Mauricio

**Texto en verde** = Completar para el 1er entregable

**Texto en azul** = Completar para el 2º entregable

**Texto en violeta** = Completar para el tercer entregable

# **RESUMEN**

La ganadería es una actividad económica que ha beneficiado en gran medida a una buena parte de la población humana durante muchos años en cuestión de consumo alimenticio. Esta industria, está en una constante evolución a causa de esta dependencia, pese a esto todavía existe un déficit en como se desarrolla esta actividad. Entre estas problemáticas se encuentran problemáticas como el control de los gases de efecto invernadero producidos por el ganado, la distribución deficiente de terreno para esta actividad, el correcto mantenimiento de salubridad en las instalaciones en los lugares dedicados a esto y por último el problema que nos trae a este trabajo que es la ineficiencia en el monitoreo de la salud 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**

Desde los inicios de la sociedad el humano ha dependido de una manera u otra de la ganadería, mayormente como fuente de alimento o la producción de derivados como el cuero. Estas prácticas que se siguen llevando hasta el día de hoy han generado una gran industria que es el sustento económico de miles de personas a lo largo del mundo, y como toda industria siempre buscará una mayor productividad a un menor costo. Es por esto que el mantenimiento de la salud del ganado es de vital importancia, pues su falla conlleva a una pérdida en la producción del animal y puede generar problemas de salud en el consumidor que a su vez lleva mayor pérdida económica, pero este proceso de vigilancia es difícil de llevar eficientemente, pues incluso con computadoras se requiere analizar una gran cantidad de datos, ahí se evidencia la utilidad que trae el uso de algoritmos de compresión en este proceso al poder alivianar parte del volumen de datos a analizar.

# **1.1. Problema**

El problema por solventar es la ineficiencia del análisis de información en la clasificación de salud animal en términos de tiempo, energía y memoria, la solución de esto acarrea consigo una mejora en términos de salubridad alimenticia y una disminución en tiempos de producción de alimentos, pues se podrá optimizar este proceso de identificación de ganado enfermo, ganado el cual puede cargar enfermedades contagiosas al ser humano y por lo tanto representar un peligro para el consumidor.

**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.

Para nuestra propuesta de solución se aplico un algoritmo de compresión con pérdidas conocido como eliminación de costuras el cual consiste en la eliminación de los puntos con menor energía de la imagen esto pues nos permitirá mantener la resolución de la imagen mientras se descarta la información menos relevante

*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.

## **3.1 Automatic monitoring system for individual dairy cows based on a deep learning framework that provides identification via body parts and estimation of body condition score.**

Este estudio se centró en el análisis del “Body Condition Score” (BCS), es decir, una métrica usada generalmente para medir el almacenamiento energético de la grasa y músculos del bovino. Para la evaluación de este puntaje se suele hacer a mano, lo cual es un proceso lento e impreciso.   
En intentos anteriores para la automatización de este proceso han fallado pues en primer lugar no mejoran notablemente la precisión o en otros casos podían generar daños al ganado.  
La solución propuesta para este problema fue la de desarrollar un framework de Deep learning entrenando la red neuronal mediante el análisis de ultrasonidos de la grasa dorsal de las vacas. Los resultados obtenidos tuvieron un coeficiente de determinación de 0,96, la precisión de las medidas fue bastante alta variando desde 0,45 hasta 0,98.[1]

**3.2 Accurate detection of lameness in dairy cattle with computer vision: A new and individualized detection strategy based on the analysis of the supporting phase**El problema presentado en este artículo es el de la imprecisión de la detección de la cojera en las vacas, los resultados de los sistemas automatizados que se han intentado implementar se ven afectados por la individualidad de cada vaca, generando diagnósticos imprecisos. Este estudio está basado en el estudio de la cojera utilizando el estado de soporte (En el ciclo de caminado de las vacas).  
Para hacer esto hicieron un algoritmo de análisis de video mezclando visión computacional de algoritmos de Deep learning. Con este método lograron clasificar correctamente el 96% de las vacas fueron correctamente clasificadas y se detectó el 93% de las pezuñas cojas con exceso de material, teniendo una precisión promedio del 87% con videos a 83,3 cuadros por segundo.[2]

## **3.3 Development and validation of a neural network for the automated detection of horn flies on cattle**

## La problemática expuesta en este trabajo es el presentado en el proceso de detección de moscas de ganado, pues este es hecho por personas lo cual lo lleva a ser ineficiente en costos y tiempo, además de llevarlo a caer en subjetividad. Implementando un método de automatización en dos fases, se hace en primer lugar una remoción de las moscas del ganado circundante de la vaca (Se va a analizar solamente las moscas de una vaca) esto se hace en un algoritmo de visión computacional y algoritmos de Deep learning, y en segundo lugar se crea un mapa de calor en base a partes de la imagen original, en el estudio se hicieron varis pruebas llegando que con una exhaustividad del 0,5 se logró una precisión del 0,9, teniendo estas dos variables (Como en la mayoría de los casos) una relación inversa.[3]

## **3.4 Detection of the respiratory rate of standing cowsby combining the Deeplab V3þsemanticsegmentation model with the phase-based videomagnification algorithm**

En este artículo se hace relevante la investigación del ratio de respiración en ganado bovino para la identificación de enfermedades y monitoreo de su salud fisiológica, pues está asociada a estrés calórico y enfermedades reproductivas.  
Esto se le buscó solución mediante el uso de tres modelos. El primero se creó un algoritmo de segmentación semántica Deeplab V3+ el cual fue creado a través de la del framework de ResNet-10, el cual fue entrenado con imágenes etiquetadas. Posteriormente un algoritmo de magnificación de video fue aplicado para observar los pequeños movimientos de la respiración de las vacas, y finalmente se usó el algoritmo de flujo óptico Lucas-Kanade para detectar la dirección de la respiración para luego pasar de un algoritmo de reconocimiento de ratio respiratorio, con esta metodología se logró una precisión entre 80% y 100%. Al aplicar este método su precisión fue de un 98,69%.[4]

## **3. MATERIALES Y MÉTODOS**

En esta sección, explicamos cómo se recogieron y procesaron los datos y, después, diferentes alternativas de algoritmos de compresión de imágenes para mejorar la clasificación de la salud animal.

## **3.1 Recopilación y procesamiento de datos**

Recogimos datos de *Google Images* y *Bing Images* divididos en dos grupos: ganado sano y ganado enfermo. Para el ganado sano, la cadena de búsqueda era "cow". Para el ganado enfermo, la cadena de búsqueda era "cow + sick".

En el siguiente paso, ambos grupos de imágenes fueron transformadas a escala de grises usando Python OpenCV y fueron transformadas en archivos de valores separados por comas (en inglés, CSV). Los conjuntos de datos estaban equilibrados.

El conjunto de datos se dividió en un 70% para entrenamiento y un 30% para pruebas. Los conjuntos de datos están disponibles en https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets .

Por último, utilizando el conjunto de datos de entrenamiento, entrenamos una red neuronal convolucional para la clasificación binaria de imágenes utilizando *Teachable Machine* de Google disponible en https://teachablemachine.withgoogle.com/train/image.

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

## En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes con pérdida. *(En este semestre, ejemplos de tales algoritmos son el tallado de costuras, el escalado de imágenes, la transformación de coseno discreto, la compresión con ondeletas y la compresión fractal).*

**3.2.1 Wavelets Compression**

Separando imágenes en secciones de diferentes frecuencias, descartando las frecuencias menos importantes (de ahí que tenga perdida) luego usando las frecuencias más prevalentes en la reconstrucción de la imagen comprimida (esta terminará con una ligera distorsión por la pérdida de datos) ejemplos de los datos descartados se da en áreas donde los valores de los píxeles tienen poco ningún cambio en su valor la compresión de datos se da a través de “wavelet transformation”, wavelets son un tipo de función para localizar una señal en el tiempo y el espacio. Wavelets grandes tienen lo general de la imagen mientras que las más pequeñas los detalles, el algoritmo cuantiza partes de la imagen y aplica la descomposición discreta de wavelets separando los componentes en bandas de los datos los más importantes otras bandas de los menos importantes guardándolos en matrices de 2 dimensiones. [5,6,7]  
  
Código QR

Descripción generada automáticamente

**3.2.2 Seam Carving**

Este algoritmo estableciendo unos caminos de menor importancia (tanto vertical como horizontalmente) y los remueve disminuyendo el tamaño de la imagen este cambio de tamaño puede generar una distorsión en los objetos presentes la importancia de los pixeles se contrasta con los valores de los pixeles vecinos como su intensidad, gradiente o “energía” siendo que los que poseen los menores valores son considerados con menor importancia.[8]  
[17]

**3.2.3 Transformada de cosenos discreta**

Este algoritmo está dado por la transformada de cosenos discreta la cual viene de la transformada de Fourier discreta siendo esta la siguiente:

Expresándola con la fórmula de Euler:

Teniendo una lista S(x) de números enteros de longitud n. Se tiene que la transformada de cosenos discreta en una dimensión es la siguiente:

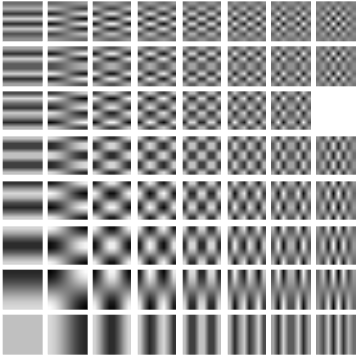
Para la compresión de imágenes con pérdida se requiere de esta transformada en dos dimensiones ya que es necesario tener una matriz de pixeles.

En primera instancia es relevante dar el concepto de imagen digital en forma de una matriz. La imagen está compuesta de pixeles los cuales son números reales organizados en la matriz de NxM (ancho x largo de la imagen) siendo la magnitud del píxel (Brillo).

Por lo tanto, se usa la siguiente formula:

Lo que está sucediendo en esta transformación con relación a los pixeles de la matriz es que cada uno de los pixeles originales cambia con relación a los pixeles contiguos a ellos en la nueva matriz creada por la transformada. Un dato relevante para mencionar es que la transformada es invertible, pero se pierde calidad en la imagen reconstruida a partir de la que ya estaba comprimida, “incluso un 85% de los coeficientes fueron descartados.” [10]

**Complejidad del algoritmo**: [9,10]

 Imagen en blanco y negro de una torre

Descripción generada automáticamente con confianza media Imagen en blanco y negro

Descripción generada automáticamente con confianza media

**3.2.4 Compresión fractal**

Es un algoritmo de compresión con pérdida de calidad usada comúnmente en imágenes en escala de grises.

Este algoritmo consiste en la división de la imagen en ***fractales*** los cuales tienen algunas de las siguientes propiedades:

* Tienen una estructura fina: Esto significa que al ampliar el fractal siempre se hallarán detalles.
* Son muy irregulares para ser descritos geométricamente.
* Se repite el mismo patrón en diferentes escalas, por lo tanto se suelen definir recursivamente.

Estos algoritmos de compresión fractal se suelen hacer de dos maneras:

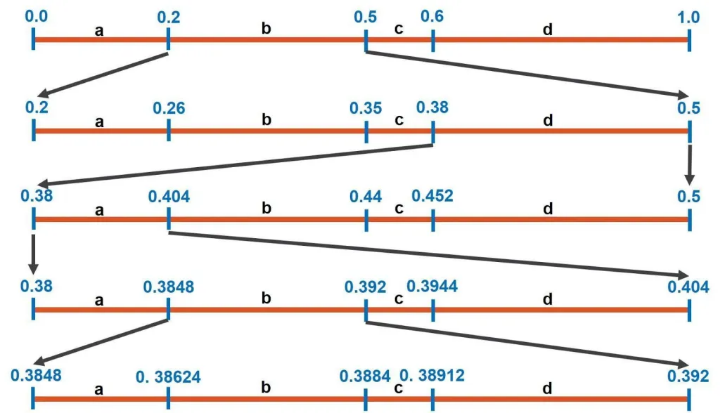
La primera, esta lo que hace es transformar una matriz de tamaño NxM a otra matriz D de tamaño axb mediante la segmentación de la matriz original en bloques, donde es el elemento es el elemento de esa nueva matriz y . N1 es en este caso   
[11]

[18]

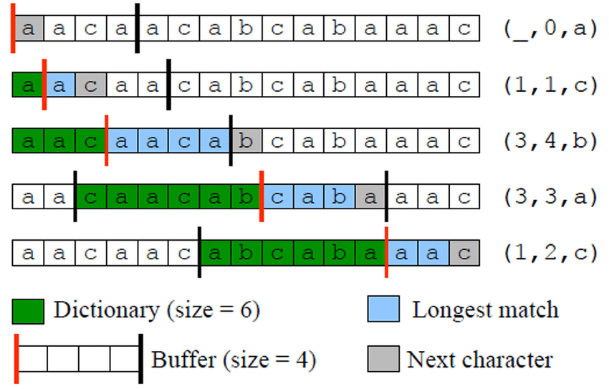
## **3.3 Alternativas de compresión de imágenes sin pérdida**

## En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes sin pérdida. *(En este semestre, ejemplos de tales algoritmos son la transformada de Borrows y Wheeler, LZ77, LZ78, la codificación Huffman y LZS).*

**3.3.1 Arithmetic coding**

Este tipo de compresión es un algoritmo que codifica una imagen usando un número entre 0.0 y 1.0, a cada color se le toma un subintervalo entre 0-1 correspondiente a su probabilidad, esta probabilidad se calcula de acuerdo a una tabla de frecuencia de cada uno de estos colores entre más común sea el color se le asigna un menor número de bits cuando se llega al final de la imagen se toma como codificación el intervalo inferior de la última reducción del intervalo.[12,13]  
[12]

**3.3.2 LZ77**

Este algoritmo analiza los datos de la imagen remplazando la información repetida con metadata relacionando secciones de la imagen idénticos o similares a puntos previos de la misma (que ya han sido reemplazados) con pequeñas fracciones de metadata indican como expandir dicha sección, posteriormente esta conjunción de datos y metadatos se serializan en una serie de bytes que luego son decodificados y descomprimidos, esto se hace a través de un pointer (que indica la cadena de coincidencias con una cadena de bits que se está buscando) y un window (que es la sección de los datos en la cual se está buscando la coincidencia de los datos) a partir de esto se le asigna una metadata a las diferentes repeticiones (esta metadata se ve determinada por la distancia del string de repetido al final de la ventana y la longitud de dicha cadena ). [14]  
[19]

**3.3.3 Algoritmo de Huffman**

Este algoritmo lo que hace es que en una secuencia de caracteres especiales recoge información sobre la frecuencia de cada carácter en la secuencia, dependiendo de esta frecuencia se le va a asignar un nuevo código de números (Entre mayor sea la frecuencia el código será de menor tamaño) para así reducir la cantidad de bits que requiere la secuencia, es decir reduce el código fuente de manera que quede el menor número de símbolos posible.

Este algoritmo consiste en la implementación del árbol de Huffman, asignando a cada carácter (En el caso de compresión de texto) la frecuencia relativa en la cual aparece, luego se organiza de menor a mayor con relación a sus frecuencias, luego se procede a sumar de izquierda a derecha cada una de las frecuencias creando así un árbol invertido donde la punta será equivalente a 1 (La suma de todas las frecuencias relativas debe ser igual a 1, es decil, el total).Teniendo en cuenta la estructura formada por el árbol descrita anteriormente se empieza a dar valores de 1 y o a las líneas transitorias de un valor a otro. Se le asigna el valor de “1” para pendientes negativas de las líneas y “0” para las pendientes positivas, luego, se hace el recorrido para llegar a la frecuencia de cada símbolo agregando los valores de ceros y unos, creando con ellos según el recorrido la nueva codificación.

Este algoritmo se ha usado por lo general para eliminar redundancia en el código.

La complejidad de este algoritmo es de donde n es el número de caracteres únicos

[15]  
Diagrama

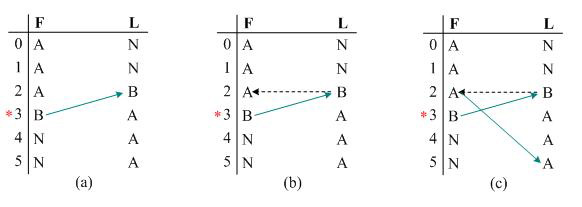
Descripción generada automáticamente Tabla

Descripción generada automáticamente

**3.3.4 Transformada de Burrows-Wheeler**

Esta transformada permite organizar y juntar caracteres que posean el mismo símbolo de una manera mas efectiva.

La ventaja que se tiene de esta transformada es que se puede codificar y decodificar sin agregar información extra, cosa que no se puede hacer normalmente con un algoritmo para ordenamiento común, ya que una vez ordenado no se puede regresar a la forma en la que se hallaba originalmente.  
El proceso que sigue esta transformada consiste en primero (En una secuencia de caracteres) ir corriendo el caracter inicial hacia la derecha, este proceso se repetirá el mismo número de veces que el tamaño de la cadena. [16]

[20]

## **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 este Proyecto se tomó la decisión de usar una matriz en el almacenamiento de la información de la imagen esto pues es la estructura más similar a como se comporta la imagen en sí, haciendo la traducción de la una a la otra más sencilla, siendo esta la principal razón por la cual se escogió pues esta simple traducción permite una manipulación más directa. Una de sus desventajas es que al recorrer y buscar los datos necesitados para el algoritmo de compresión escogido (el “camino” con menor energía) es bastante complejo en términos de tiempo de ejecución pues este proceso es de orden O(n\*m) siendo n y m el numero de filas y columnas respectivamente lo cual hace que al implementar el algoritmo este se demore un largo periodo de tiempo al ejecutarlo

**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**

El algoritmo con Perdida que se eligió para el proyecto es el de eliminación de costuras el cual consiste en el cálculo y posterior eliminación de lo puntos menos relevantes de una imagen, para esto se aplica un filtro Sobel en la imagen para obtener las derivadas parciales en ambos ejes de la imagen para luego sumar sus valores absolutos posteriormente aplicando conceptos de programación dinámica se buscan los caminos de menor energía desde la parte superior de la imagen hasta la inferior el cual procederá a ser eliminado, este proceso se repite una cantidad de veces equivalente al porcentaje que quieras cortar de la imagen

Tiene la ventaja de tener una cierta “conciencia” de los objetos lo cual significa que alterara lo menos posible los objetos ahí presentes centrándose mas en los fondos, este algoritmo tiene como inconveniente en primer lugar su lentitud pudiendo llegar a tomar varios minutos en comprimir una sola imagen, además de esto si se eliminan demasiadas partes de la imagen esta va a empezar a deformarse haciendo que la compresión no siempre sea tan notable a comparación de otros algoritmos.

**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).*

**4.4 Criterios de diseño del algoritmo**

Explica por qué el algoritmo fue diseñado de esa manera. Use un criterio objetivo. Los criterios objetivos se basan en la eficiencia, que se mide en términos de tiempo y consumo de memoria. Ejemplos de criterios no objetivos son: "Estaba enfermo", "fue la primera estructura de datos que encontré en Internet", "lo hice el último día antes del plazo", etc. Recuerde: Este es el 40% de la calificación del proyecto.

**5. RESULTADOS**

**5.1 Evaluación del modelo**

En esta sección, presentamos algunas métricas para evaluar el modelo. La exactitud es la relación entre el número de predicciones correctas y el número total de muestras de entrada. La precisión es la proporción de estudiantes exitosos identificados correctamente por el modelo a estudiantes exitosos identificados por el modelo. Por último, sensibilidad es la proporción de estudiantes exitosos identificados correctamente por el modelo a estudiantes exitosos en el conjunto de datos.

**5.1.1 Evaluación del conjunto de datos de entrenamiento**

A continuación presentamos las métricas de evaluación del conjunto de datos de entrenamiento en la Tabla 3.

|  |  |
| --- | --- |
|  | ***Conjunto de datos de entrenamiento*** |
| *Precisión* | 0.02 |
| *Precisión* | 0.03 |
| *Recordar* | 0.01 |

## **Tabla 3.** Evaluación del modelo de clasificación de imágenes con el conjunto de datos de entrenamiento.

**5.1.2 Evaluación del conjuntos de datos de prueba**

A continuación presentamos las métricas de evaluación del conjunto de datos de prueba, en la Tabla 4, sin compresión y, en la Tabla 5, con compresión.

|  |  |
| --- | --- |
|  | ***Conjunto de datos de prueba*** |
| *Exactitud* | 0.01 |
| *Precisión* | 0.012 |
| *Sensibilidad* | 0.013 |

## **Tabla 4.** Evaluación del modelo de clasificación de imágenes, con el conjunto de datos de prueba, sin compresión.

|  |  |
| --- | --- |
|  | ***Conjunto de datos de prueba*** |
| *Exactitud* | 0.001 |
| *Precisión* | 0.0012 |
| *Sensibilidad* | 0.0013 |

## **Tabla 5.** Evaluación del modelo de clasificación de imágenes, con el conjunto de datos de prueba, con compresión.

**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* | 100.2 s | 12.4 MB |
| *Descompresión* | 800.1 s | 12.4 MB |

## **Tabla 6:** Tiempo deejecución de los algoritmos *(Por favor, escriba el nombre de los algoritmos, por ejemplo, tallado de costuras y LZ77)* para diferentes imágenes en el conjunto de datos.

## **5.3 Consumo de memoria**

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

|  |  |  |
| --- | --- | --- |
|  | ***Consumo promedio de memoria (MB)*** | ***Tamaño promedio del archivo (MB)*** |
| Compresión | 634 MB | 3.12 MB |
| Descompresión | 9 MB | 878.12 MB |

## **Tabla 7:** Consumopromedio de memoria de todas las imágenes del conjunto de datos, tanto para la compresión como para la descompresión.

## Para medir el consumo de memoria, deberían usar un generador de perfiles. Uno muy bueno para Java es VisualVM, desarrollado por Oracle, http://docs.oracle.com/javase/7/docs/technotes/guides/visualvm/profiler.html. Para Python, usa el *C profiler*.

## **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 | 1:23 | 1:34 |

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

## **6. DISCUSIÓN DE LOS RESULTADOS**

Explique los resultados obtenidos. ¿Son la exactitud, la precisión y la sensibilidad apropiadas para este problema? ¿El modelo está sobreajustado? ¿Es apropiado el consumo de memoria y el consumo de tiempo? ¿Es la relación de compresión apropiada? ¿Cambia la compresión significativamente la exactitud con el conjunto de datos de la prueba? *(En este semestre, según los resultados, ¿puede la compresión mejorar la clasificación de la salud animal en el contexto del PLF? )*

**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?

# **RECONOCIMIENTOS**

Identifique el tipo de reconocimiento que quiere escribir: para una persona o para una institución. Considere las siguientes pautas: 1. El nombre del profesor no se menciona porque es un autor. 2. No debe mencionar sitios web de autores de artículos que no ha contactado. 3. Debe mencionar estudiantes, profesores de otros cursos que le hayan ayudado.

Como ejemplo: Esta investigación fue apoyada/parcialmente apoyada por [Nombre de la Fundación, Donante, Beca].

Agradecemos la asistencia con [técnica particular, metodología] a [Nombre Apellido, cargo, nombre de la institución] por los comentarios que mejoraron enormemente el manuscrito o la codificación del algoritmo.

# **REFERENCIAS**

La referencia se obtiene usando el formato de referencia ACM. Lea las directrices de ACM en <http://bit.ly/2pZnE5g>

A modo de ejemplo, consideremos estas dos 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.

Por favor, quite las referencias de arriba, son sólo un ejemplo.

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