Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

TC3007C

Inteligencia artificial avanzada para la ciencia de datos II

Entendimiento de Datos: Big Data

Autores:

A01368818 Joel Sánchez Olvera

A01661090 Juan Pablo Cabrera Quiroga

A01704076 Adrián Galván Díaz

A01708634 Carlos Eduardo Velasco Elenes

A01709522 Arturo Cristián Díaz López

TC3007C.501

Inteligencia artificial avanzada para la ciencia de datos II

Fecha:

14 de Octubre del 2024

Introducción

Este documento aborda el desarrollo de un proyecto enfocado en la implementación de inteligencia artificial avanzada para la detección y clasificación de posiciones de vacas en imágenes, un desafío planteado por CAETEC. En el contexto de la ciencia de datos, este tipo de proyectos demanda tanto el procesamiento de grandes volúmenes de información como el uso de modelos de aprendizaje profundo que puedan optimizar tareas repetitivas y mejorar la precisión en el análisis de imágenes. Para lograr estos objetivos, se emplean frameworks de modelos para machine learning, así como técnicas de limpieza y preparación de datos que aseguran la calidad y relevancia del conjunto de datos procesado.

El trabajo se divide en diversas etapas, cada una de las cuales aborda una parte fundamental en el desarrollo de un sistema de machine learning. Se inicia con una descripción de las herramientas y tecnologías empleadas, justificando su uso y relevancia. Posteriormente, se presenta el modelo de almacenamiento y procesamiento de datos utilizado, junto con una propuesta de escalabilidad en el futuro. La tercera sección se centra en la extracción, limpieza y carga de los datos, destacando la importancia de la organización y preparación de las imágenes antes de su análisis. Adicionalmente, se aplica un esquema de validación k-fold cross-validation para evaluar el desempeño del modelo, evitando problemas de sobreajuste y garantizando una mayor robustez en la predicción. Por último, se analiza la viabilidad de emplear un enfoque orientado a Big Data en caso de que el proyecto escale, considerando el potencial de crecimiento en el número de imágenes y otros datos asociados.

Este proyecto representa una oportunidad para aplicar técnicas avanzadas de inteligencia artificial en el análisis de imágenes y muestra cómo estas tecnologías pueden optimizar tareas en industrias que requieren un procesamiento eficiente de datos visuales. A lo largo del documento, se examinan cada uno de estos aspectos, proporcionando una visión detallada del proceso de implementación y justificación de cada decisión técnica.

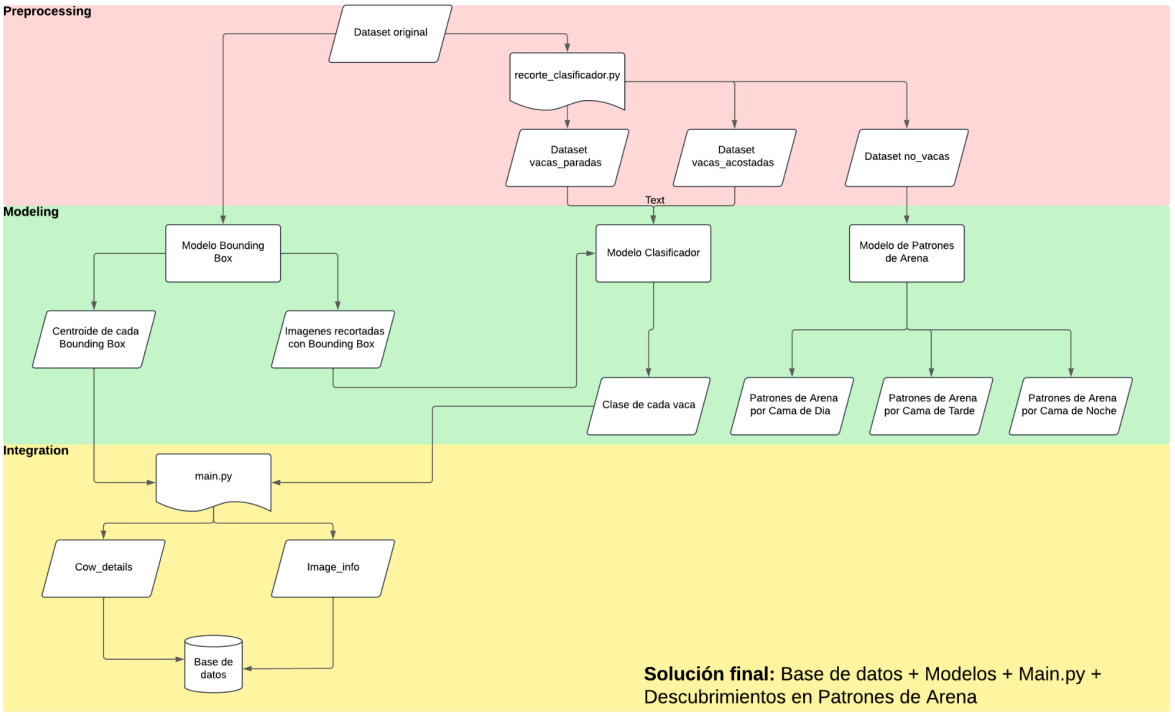
Herramientas y tecnologías que se usarán para trabajar con los datos (Justificación)

Fase	Objetivo	Herramientas	Justificación
Extracción	Obtener todos los datos posibles garantizando la integridad y calidad de los datos extraídos para satisfacer los requerimientos iniciales del proyecto.	- One Drive	Los datos originales dados por CAETEC estaban dentro de un directorio de OneDrive, el cual ofrece alta seguridad mediante permisos personalizados , además de facilitar su actualización en caso de ser necesario gracias a su sincronización automática.
Transformación	Aplicar procesos de limpieza, normalización y de datos para estandarizar los formatos y asegurar la coherencia, haciendo que los datos sean estructurados y consistentes para el análisis o modelado posterior.	- Python - Roboflow	**Bounding Box** Para el bounding box, se utilizó la herramienta de Roboflow ya que ésta provee una interfaz gráfica para hacer el etiquetado de las imágenes de manera más eficiente. Esto nos ayudó para crear las bounding boxes de las vacas en cada una de las imágenes del dataset original, lo cual hizo el proceso mucho más eficiente y esa fue la razón principal por la cual decidimos utilizar Roboflow. **Clasificador** Para el modelo clasificador modificamos el tamaño de la imagen a 224 x 224 para una normalización estándar de las mismas. Así mismo se estandarizó el brillo de la imagen convirtiendo los valores de los píxeles a un tipo de dato float32 y después dividiéndolos entre 255 para que queden en un rango entre 0 y 1. Esto es común en el preprocesamiento de imágenes, ya que ayuda a mejorar el rendimiento y la estabilidad de muchos modelos de machine

			learning.
Load (Carga)	Almacenar los datos transformados en un sistema de destino de manera eficiente y segura, asegurando su disponibilidad para consultas y análisis, y minimizando la latencia en el acceso a la información final.	- Google Drive	Usamos Google Drive para Cargar los datos ya que provee una buena seguridad para proteger los datos y controlar los accesos, así como poder tener los datos accesibles desde cualquier parte y poder usarlos para trabajar en los modelos, Google Drive nos da una manera eficiente y segura para guardar los datos ya procesados y que puedan ser usados por el equipo.
Modelado	Modelar correcta y eficientemente 2 modelos que trabajen con los datos, un modelo de bounding box y un modelo de clasificación, ambos trabajando con los datos proporcionados por el socio , con el fin de poder identificar y clasificar las vacas y sus posiciones dentro de la imagen.	- Tensor Flow - Pytorch - YOLOv5 (modelo pre-entenado)	**Bounding Box** Para el modelo de bounding Box, implementamos una arquitectura de redes neuronales utilizando PyTorch, haciendo uso de YOLOv5, una arquitectura de detección de objetos en tiempo real, la cual usamos para identificar y localizar las vacas en las imágenes, utilizamos YOLOv5 ya que es reconocido por su alta precisión y velocidad, haciéndolo ideal para aplicaciones que requieren un procesamiento eficiente de imágenes, como lo requería el proyecto. Gracias a YOLOV5 el modelo puede detectar múltiples vacas en una sola imagen y generar cuadros delimitadores (*bounding boxes*) que indican sus posiciones exactas así como sus centroides. **Clasificador** Para el modelo de clasificación, utilizamos una arquitectura de red neuronal convolucional (CNN) implementada en PyTorch. Este modelo se entrena para analizar imágenes de vacas y clasificarlas según su posición, si se encuentran acostadas o paradas.

			La elección de PyTorch se debe a su flexibilidad y facilidad de uso, lo que facilita el modelado y el ajuste de hiper parámetros durante el desarrollo del modelo. Además, PyTorch nos da una integración eficiente con GPU, acelerando el proceso de entrenamiento y haciendo el manejo de grandes volúmenes de datos más efectivo.
Análisis de patrones en la arena	Determinar la existencia de patrones repetitivos o estructuras organizadas en la superficie de las camas de arena que pudieran afectar el comportamiento de descanso del ganado	- Python (OpenCV, NumPy, SciPy, Matplotlib, Seaborn)	Para el análisis de patrones se implementaron múltiples técnicas de procesamiento de imágenes y análisis de textura: 1) Análisis de textura mediante GLCM para evaluar características como contraste, homogeneidad y correlación. 2) Detección de bordes usando métodos Sobel, Canny y Laplaciano para identificar estructuras. 3) Análisis de periodicidad y escala para detectar patrones repetitivos. 4) Mapeo de direcciones de gradiente para estudiar orientaciones preferentes. Esta combinación de técnicas proporcionó un análisis comprehensivo que permitió evaluar la superficie desde múltiples perspectivas, asegurando la detección de cualquier patrón significativo que pudiera afectar al ganado.
Visualización de los datos	Obtener hallazgos significativos a partir de la base de datos generada	- Tableau	Para visualizar y analizar los datos, utilizamos Tableau debido a su capacidad para transformar datasets complejos en gráficos interactivos y dashboards intuitivos, permitiendo explorar y filtrar los datos de manera dinámica. Tiene una interfaz amigable y un gran soporte para grandes volúmenes de datos.

Diagrama de Flujo de Solución



Explicación de las etapas del proceso.

1. Preprocessing

En esta sección, se realiza la preparación inicial de los datos a partir de un dataset original.

- **Dataset original:** Punto de partida que contiene imágenes de vacas y otros elementos.
- **`recorte_clasificador.py`:** Un script que se encarga de dividir el dataset original en tres conjuntos:
  - **`Dataset vacas_paradas`:** Imágenes clasificadas de vacas en posición de pie.
  - **`Dataset vacas_acostadas`:** Imágenes clasificadas de vacas acostadas.
  - **`Dataset no_vacas`:** Imágenes que no contienen vacas.

Esta fase segmenta el dataset original para facilitar su uso en modelos específicos.

2. Modeling (Modelado)

En esta fase se implementan tres modelos diferentes, cada uno con un propósito particular:

1. **Modelo Bounding Box:**
    - Detecta regiones específicas en las imágenes que contienen vacas y genera:
        - **Centroides de cada Bounding Box:** Coordenadas del centro de las cajas delimitadoras de las vacas detectadas.
        - **Imágenes recortadas:** Fragmentos de la imagen original basados en las Bounding Boxes.
2. **Modelo Clasificador:**
    - Clasifica las vacas detectadas en función de su estado (de pie o acostadas).
    - **Clase de cada vaca:** Proporciona la etiqueta correspondiente a cada vaca detectada.
3. **Modelo de Patrones de Arena:**
    - Analiza patrones de arena en las camas donde descansan las vacas.
    - Divide los resultados en diferentes periodos del día:
        - Patrones de Arena por Cama de Día
        - Patrones de Arena por Cama de Tarde
        - Patrones de Arena por Cama de Noche

3. Integration (Integración)

Esta fase combina los resultados de los modelos y organiza los datos en una estructura utilizable:

- **main.py:** Script principal que gestiona la integración de los datos procesados.
    - **Cow_details:** Información detallada de cada vaca, incluyendo su posición (de pie o acostada) y ubicación (centroides).
    - **Image_info:** Información asociada a las imágenes procesadas, como los patrones de arena detectados.
- Ambos resultados se almacenan en una **base de datos** para su análisis y acceso posterior.

Solución Final

El resultado del proceso completo es una solución integrada que incluye:

- **Base de datos:** Contiene la información estructurada sobre las vacas identificadas, los centroides encontrados y estado de la vaca en la hora de la muestra..
- **Modelos:** Bounding Box y modelo clasificador implementados y configurados para procesar nuevas imágenes.
- **main.py:** Gestiona la ejecución del flujo de trabajo de integración.

Flujo General de los Datos

1. El **dataset original** se segmenta en subconjuntos utilizando el script **recorte_clasificador.py.**
2. Los subconjuntos se procesan por separado en tres modelos:
    - **Bounding Box** para detección y recorte.
    - **Clasificador** para etiquetar las vacas según su posición.
    - **Patrones de Arena** para analizar los patrones en las camas.
3. Los resultados de los modelos se integran mediante `main.py` y se almacenan en una base de datos para consultas y análisis.

Diagrama de despliegue

El diagrama describe el flujo de trabajo para el despliegue, procesamiento y exportación de resultados del modelo basado. A continuación, se detallan los pasos clave representados en el esquema:

1. **Almacenamiento (Google Drive):** El proyecto utiliza actualmente Google Drive como repositorio de datos para las 9634 imágenes que componen el conjunto de datos.
2. **Carga de Imágenes:** Las imágenes almacenadas en Google Drive se cargan a un Servidor de Procesamiento, donde se realizan las etapas posteriores de análisis y predicción.
3. **Servidor de Procesamiento (Python Environment):** En este servidor, se utiliza un entorno de Python configurado para ejecutar modelos de aprendizaje profundo. Aquí, las imágenes son preprocesadas y enviadas al modelo correspondiente.
4. **Modelos (TensorFlow/PyTorch):** El procesamiento de las imágenes ocurre en esta etapa, donde los modelos implementados en TensorFlow o PyTorch clasifican y analizan las imágenes según los objetivos de cada modelo.
5. **Almacenamiento de Resultados (Base de Datos SQL):** Los resultados obtenidos del modelo son almacenados en una base de datos SQL. Este paso asegura la organización y disponibilidad de los datos para su consulta y análisis posterior.
6. **Exportación de Resultados (CSV):** Los resultados almacenados en la base de datos SQL pueden ser exportados en formato CSV para facilitar su interpretación, compartirlos con otros equipos o realizar análisis adicionales en herramientas externas.

Almacenamiento utilizado

Dataset

Actualmente, el dataset se encuentra alojado en Google Drive como repositorio de datos para las 9,634 imágenes que componen el conjunto de datos. Cada imagen tiene un tamaño promedio de entre 600 y 650 KB, lo que hace que el almacenamiento requiera una solución que permita tanto la organización como la accesibilidad constante para análisis y procesamiento.

Google Drive fue elegido por las siguientes razones principales:

- **Facilidad de Uso**: Su interfaz intuitiva y ampliamente conocida facilita el acceso y manejo de los datos, lo que reduce la curva de aprendizaje para el equipo.
- **Colaboración**: Permite un acceso sencillo y rápido desde cualquier ubicación, lo cual es esencial en un entorno de trabajo colaborativo como el de este proyecto.
- **Configuración Simple**: A diferencia de soluciones más robustas como Amazon S3, la configuración de Google Drive no requiere conocimientos técnicos avanzados, lo que permitió una implementación inmediata y eficiente.

Resultados

Los resultados arrojados por el modelo deben ser almacenados en una base de datos relacional, asegurando la integridad y consistencia de la información registrada. Cada detección realizada se traduce en un registro estructurado que incluye detalles como las coordenadas del *bounding box*, el centroide, la confianza en la predicción y la postura identificada (por ejemplo, *vaca acostada* o *vaca de pie*). Esta información se distribuye en dos tablas principales:

1. **Tabla `image_info`:**
   Encargada de registrar datos generales sobre las imágenes procesadas, como el número total de vacas detectadas en cada imagen y el *timestamp* de procesamiento.
2. **Tabla `cow_details`:**
   Diseñada para almacenar detalles específicos de cada vaca detectada en una imagen, vinculándolos mediante una clave foránea a la tabla `image_info`. Aquí se registran el centroide, las coordenadas del *bounding box*, la postura detectada y el nivel de confianza de la predicción realizada por el modelo.

En el desarrollo de este proyecto, se decidió montar la base de datos de manera **local**, directamente en la Raspberry Pi. Esta decisión responde a las necesidades específicas del sistema, que busca operar de forma eficiente, autónoma y segura en un entorno controlado. Al ser una solución diseñada para procesar datos en tiempo real y en un espacio limitado, priorizamos factores como la **baja latencia**, **independencia de red**, y el **control completo de los datos**, aspectos que resultan fundamentales para garantizar la confiabilidad del sistema.

Este enfoque no solo permite optimizar los recursos del hardware disponible, sino también mitigar posibles riesgos de seguridad asociados con el manejo de datos sensibles en redes externas, brindando una solución robusta y económica para el contexto del proyecto. A continuación, se muestra una tabla comparativa que permite identificar las ventajas y desventajas de ambas alternativas.

Criterio	Base de Datos Local	Base de Datos Remota
Proximidad de Datos	Los datos y el almacenamiento interactúan en el mismo sistema, reduciendo la latencia y tiempos de transferencia de datos.	Los datos deben transferirse a través de la red, aumentando la latencia.
Independencia de Red	No depende de una conexión a internet para funcionar.	Requiere conexión constante a la red, lo que puede ser un punto de falla.
Seguridad	Mayor control sobre los datos, ya que no se exponen a redes externas.	Riesgo de brechas de seguridad si no se implementan medidas adecuadas.
Costo	Sin costos adicionales, ya que no se usan servicios externos.	Puede incurrir en costos recurrentes por almacenamiento y transferencia.
Escalabilidad	Limitada por el hardware local (espacio y procesamiento).	Fácilmente escalable en plataformas como AWS, Google Cloud o Azure.
Configuración	Requiere configuración inicial más detallada, pero	Simplificada, pero limitada por las opciones del

	control total.	proveedor.
Accesibilidad	Accesible solo desde dispositivos conectados directamente o conexiones a través de un cliente remoto usando SSH.	Accesible desde cualquier lugar con conexión a internet.
Resiliencia	Depende del hardware local, lo que puede ser un problema en caso de fallas.	Generalmente alta, ya que los proveedores ofrecen redundancia y copias de seguridad.
Flexibilidad	Completamente personalizable según las necesidades del proyecto.	Restringida por las configuraciones y políticas del proveedor.
Riesgo de dependencia	Sin dependencia de terceros.	Alta dependencia del proveedor de servicios y sus términos.

¿Por qué elegir un modelo de almacenamiento local?
Este modelo es ideal para el proyecto trabajado ya que ofrece una solución autónoma, sin dependencia de terceros ni dependencia a una red. Además, la latencia es menor al trabajar en un único sistema integrado.

Consideraciones sobre Amazon S3

Si bien una base de datos local satisface las necesidades actuales del proyecto, en escenarios futuros que demanden mayor escalabilidad y robustez, Amazon S3 podría ser una solución viable. Amazon S3 (Simple Storage Service) es ampliamente reconocido por ofrecer:

- **Escalabilidad**: Ideal para manejar grandes volúmenes de datos, como imágenes o videos.
- **Alta Disponibilidad y Durabilidad**: Asegura que los datos estén accesibles desde cualquier parte del mundo en todo momento.
- **Flexibilidad y Seguridad**: Su diseño modular permite integraciones avanzadas y niveles de seguridad robustos.

Decisión Actual y Futuras Iteraciones

A pesar de estas ventajas, no consideramos necesario utilizar Amazon S3 para la solución actual, ya que el proyecto no requiere un enfoque basado en Big Data (la justificación a esta decisión se encuentra en la última sección de este documento). Google Drive y una base de datos local cumplen con los requerimientos de accesibilidad y simplicidad que el equipo de desarrollo necesita.

La siguiente sección presenta una breve guía que cubre los pasos necesarios para montar un bucket de Amazon S3 en futuras iteraciones del proyecto, en caso de que las necesidades de almacenamiento o procesamiento escalen significativamente.

Sin embargo, estamos abiertos a considerar Amazon S3 para futuras iteraciones del proyecto. A continuación, presentamos un esquema general sobre cómo podríamos implementar Amazon S3 en el futuro, en caso de que las necesidades de almacenamiento o procesamiento escalen significativamente.

<u>Pasos para implementar S3 en el proyecto</u>

1. **Crear un Bucket en S3**
    a. Ir a la sección de S3 y crear un nuevo bucket donde se almacenarán las imágenes del proyecto.
    b. Definir los permisos de acceso para controlar quién puede subir y acceder a las imágenes, asegurando que solo los miembros del equipo autorizado tengan acceso.

2. **Cargar los datos**
    a. Subir las 9634 imágenes existentes al bucket de S3. Esto se puede hacer manualmente a través de la consola de AWS o programáticamente utilizando herramientas como la AWS CLI (Command Line Interface) o el SDK de Python, Boto3.
    b. Para el futuro, se podría automatizar la carga de nuevas imágenes utilizando scripts en Python que carguen automáticamente cualquier nuevo archivo de imagen al bucket.

3. **Configurar permisos y políticas**
    a. Definir políticas de bucket para establecer restricciones de acceso. Por ejemplo, el acceso a los datos podría estar restringido por IP o por los roles de los usuarios.

b. Habilitar el cifrado automático de los datos para proteger la información sensible durante la transmisión y almacenamiento.

4. **Integrar el acceso a S3 desde el código del proyecto**
    a. Utilizar la biblioteca Boto3 en Python para interactuar con el bucket de S3 directamente desde el código. Esto permitirá que el equipo descargue, suba o procese las imágenes de manera dinámica durante la ejecución de los scripts de procesamiento de datos o entrenamiento de los modelos.
        - import boto3 s3 = boto3.client('s3') bucket_name = 'nombre-del-bucket'
        - Ejemplo para descargar una imagen desde el bucket: s3.download_file(bucket_name, 'nombre_de_la_imagen.jpg', 'ruta_local

5. **Monitorización y optimización**
    a. Usar servicios adicionales de AWS como CloudWatch para monitorear el uso del bucket y recibir alertas en caso de cualquier actividad inusual.
    b. Configurar el versionado de objetos en el bucket de S3 para llevar un registro de las diferentes versiones de las imágenes en caso de que se requiera recuperar una versión anterior.

El **diagrama de despliegue** se puede consultar como anexo al final de este documento.

Extracción, limpieza y carga del conjunto de datos

El equipo ha seleccionado 4004 imágenes para realizar un análisis más eficiente. Las imágenes se han dividido equitativamente entre los miembros del equipo para su clasificación, lo que garantiza un procesamiento eficiente y uniforme de los datos. Durante esta etapa, se recortaron manualmente las imágenes para centrarse en las camas de las vacas, etiquetándolas en tres categorías: cama vacía, vaca parada y vaca acostada. Esta limpieza y preparación de los datos asegura que el modelo solo utilice la información relevante, evitando el procesamiento innecesario de áreas de la imagen que no aportan valor al análisis.

Para realizar la clasificación mencionada, hicimos uso de un script de python, el cual recorta las imágenes y nos permite eficientar el proceso de clasificación mostrando la imagen y permitiendo escoger una carpeta específica de cada clase en la cual se guardará la imagen recortada. El script puede ser consultado aquí en la carpeta de source.

Adicionalmente para nuestro proyecto, tenemos considerado implementar un modelo para detectar las vacas en las imágenes de manera automática, en lugar de hacerlo de manera manual. Para este modelo de object detection, necesitamos utilizar otro dataset que tenga las etiquetas de los diferentes bounding boxes de las vacas.

Para hacer de este proceso más eficiente (en cuestión de costo), utilizamos la herramienta Robo Flow, que es un software que utiliza inteligencia artificial para identificar las vacas y las encierra en bounding boxes. Utilizamos esta herramienta porque nos iba a salir más barato que etiquetar estas imágenes de manera manual.

Separación del conjunto de datos (Esquema k-fold cross validation)

El k-fold cross-validation es una técnica de validación utilizada en machine learning para evaluar el rendimiento de un modelo de manera más robusta y confiable. Sirve para garantizar que el modelo no esté sobreajustado (overfitting) ni subajustado (underfitting) y que generalice bien en datos nuevos.

Se utiliza una división en k partes: El dataset se divide en k subconjuntos (folds) del mismo tamaño y utiliza un proceso iterativo:

- En cada iteración, se entrena el modelo utilizando k-1 folds como conjunto de entrenamiento.
- El fold restante (que no se usó en el entrenamiento) se utiliza como conjunto de validación para evaluar el modelo.
- Este proceso se repite k veces, rotando los folds para que cada subconjunto se utilice una vez como validación.

Promedio de resultados: Después de realizar las k iteraciones, se calculan métricas como la precisión, el error, etc., promediando los resultados obtenidos en cada fold. Esto proporciona una estimación más confiable del rendimiento del modelo.

El script que nosotros utilizamos para hacer el k-fold cross validation se encuentra en nuestro repositorio de Github en la carpeta de source. El split del dataset se encuentra en la carpeta de dataset.

¿Es necesario utilizar un enfoque orientado a Big Data? (Justificación)

En este proyecto específico, el conjunto de datos consta de 9634 imágenes, por lo cual nosotros consideramos que **no es necesario darle un enfoque de Big Data al proyecto**. Aunque este volumen de datos no justifica por sí solo un enfoque orientado a Big Data, el uso de tecnologías avanzadas puede ser relevante si el proyecto se expande. Si en el futuro se incluyen datos adicionales, como videos o sensores en tiempo real, la cantidad de información podría aumentar significativamente, requiriendo un procesamiento distribuido. En tal caso, herramientas como PySpark y Hadoop permitirían manejar este incremento de volumen y facilitar el análisis paralelo de grandes cantidades de datos.

Además del volumen de imágenes que tenemos, otros principios del Big Data deben considerarse:

1. **Variedad**

   No cumple con criterio de Big Data

   Actualmente, el proyecto se centra únicamente en imágenes, esto significa que no incorpora ni integra diferentes tipos de datos, lo que limita su alcance en términos de variedad, ya que no existe la necesidad de procesar múltiples formatos o tipos de datos simultáneamente.

   Sin embargo, el crecimiento del volumen de los datos a procesar y tomando en cuenta que existe un claro potencial de expansión hacia la incorporación de diversos tipos de datos, como vídeos, sensores ambientales, registros de actividad, entre otros. Lo que abriría nuevas posibilidades de análisis más complejos y completos. Este tipo de integración requeriría adoptar un enfoque de Big Data para manejar esta variedad de formatos y fuentes de manera cohesiva.

2. **Velocidad**:

   No cumple con criterio de Big Data

   Actualmente, las imágenes se procesan de manera estática y no en tiempo real. La baja frecuencia de generación y la ausencia de flujos continuos no justifican un enfoque Big Data, ya que no hay urgencia en el análisis.

Sin embargo, el proyecto tiene potencial de expansión hacia la integración de flujos de video en tiempo real y sensores de actividad de las vacas. En este escenario futuro, la velocidad de procesamiento se volvería crítica, requiriendo un enfoque de Big Data para procesar y analizar datos en tiempo real, mejorando así la capacidad de respuesta y toma de decisiones inmediatas en el rancho.

3. **Veracidad**:

   Sí cumple con criterio de Big Data

   El proyecto sí cumple con el criterio de veracidad, esto se debe a que las imágenes del dataset fueron tomadas directamente en el campo de trabajo, garantizando que representen fielmente las condiciones reales donde operará el modelo. Además, el etiquetado y la revisión manual de los datos permitieron detectar y corregir inconsistencias, asegurando su calidad y confiabilidad.

4. **Valor**:

   Sí cumple con criterio de Big Data

   Esto se debe a que el valor principal del proyecto está directamente alineado con la necesidad principal del socio: optimizar el uso de camas y mejorar la gestión de recursos. El modelo desarrollado proporciona una solución práctica al permitir un monitoreo preciso de las actividades de las vacas, ayudando a identificar patrones de comportamiento y posibles mejoras en la distribución y uso de las camas. Aunque actualmente el volumen de datos es manejable sin Big Data, el proyecto ya está generando valor al responder a la necesidad específica del socio.

En resumen, presentamos la siguiente tabla donde identificamos si necesitamos un acercamiento Big Data en cada uno de los criterios de Big Data:

Volumen	Variedad	Velocidad	Veracidad	Valor
❌	❌	❌	✅	✅

Al solo contar con 2 de los 5 criterios de Big Data, nosotros consideramos que **no se necesita un enfoque de Big Data.**

Conclusión

En este proyecto, hemos determinado que no es necesario adoptar un enfoque de Big Data debido a que solo cumplimos con 2 de los 5 criterios establecidos (veracidad y valor). Aunque el volumen, la variedad y la velocidad de los datos no justifican la necesidad de herramientas como Hadoop o PySpark, hemos optado por utilizar tecnologías avanzadas como PyTorch y TensorFlow.

Estas herramientas son ampliamente utilizadas en proyectos de Big Data debido a su capacidad para manejar modelos complejos y procesar grandes volúmenes de información de manera eficiente.

A pesar de no trabajar con un problema de Big Data, PyTorch y TensorFlow ofrecen ventajas significativas:

- **Optimización de Modelos**: Permiten un entrenamiento eficiente de modelos de aprendizaje profundo incluso con volúmenes de datos más pequeños.
- **Escalabilidad**: Si en el futuro el volumen de datos o su complejidad crecen, estas herramientas pueden adaptarse fácilmente a contextos más exigentes sin necesidad de cambiar de tecnología.
- **Desempeño Computacional**: Ambas bibliotecas aprovechan hardware como GPUs para acelerar el procesamiento de datos, incluso cuando el volumen no alcanza los niveles de Big Data.
- **Soporte Comunitario y Flexibilidad:** Cuentan con un robusto ecosistema de desarrolladores, lo que facilita la implementación de nuevas funcionalidades y la resolución de problemas.

En resumen, aunque no consideramos que este proyecto requiere un enfoque de Big Data, hemos elegido tecnologías que no solo cubren las necesidades actuales, sino que también permiten preparar el proyecto para una posible expansión.

En caso de que en el futuro se integren flujos de video, datos de sensores en tiempo real u otros tipos de datos, estas herramientas pueden escalar para manejar volúmenes y velocidades mayores, abriendo la puerta a un enfoque Big Data si es necesario. Esta estrategia asegura que el proyecto sea eficiente y flexible, alineándose con los objetivos presentes sin perder de vista las posibles necesidades futuras.

Anexos

Link del repositorio de GitHub: https://github.com/JP-coder2000/cattle-segmentation
Diagrama de despliegue: Despliegue Cattle Segmentation.png