Universidad Internacional de Valencia

Detección de satélites en órbitas geoestacionarias mediante imágenes de baja resolución

Titulación:
Máster en Inteligencia Artificial
Curso académico 2022-2023

Alumno/a: Pérez Pérez, Marc
D.N.I: 26746406M

Director/a de TFM: Cesar Augusto Guzman Alvarez

Convocatoria:

Primera

De:
Planeta Formación y Universidades

17 Agosto 2023

Índice

Anexo de imágenes

1. Introducción

La inteligencia artificial (IA) ha experimentado avances significativos en las últimas décadas, abriendo un amplio abanico de posibilidades en diversos campos. Uno de los campos de aplicación en constante crecimiento es el ámbito espacial, donde la IA desempeña un papel fundamental en el análisis de imágenes y datos astronómicos. En este contexto, la detección de satélites geoestacionarios en imágenes del cielo nocturno se ha convertido en un desafío relevante que puede aprovechar los avances en técnicas de visión por computadora y aprendizaje automático.

Contexto y relevancia

En el ámbito de las tecnologías espaciales, los satélites en órbita geoestacionaria (GEO) desempeñan un papel fundamental en una variedad de aplicaciones. Estos satélites se encuentran en una órbita especial denominada órbita geoestacionaria, donde su velocidad de rotación coincide con la velocidad de rotación de la Tierra. Esto permite que los satélites GEO se mantengan aparentemente estacionarios en relación a un punto fijo en la superficie terrestre.

La relevancia de los satélites GEO radica en su capacidad para proporcionar servicios de comunicación, observación de la Tierra y otras aplicaciones espaciales. En el ámbito de las comunicaciones, estos satélites se utilizan para la transmisión de señales de televisión, radio, telefonía móvil y servicios de internet de banda ancha en todo el mundo. Su posición estacionaria sobre un punto específico de la Tierra permite una cobertura continua y una comunicación confiable en vastas áreas geográficas.

Además de las comunicaciones, también juegan un papel crucial en la observación de la Tierra y la meteorología. Estos satélites proporcionan imágenes y datos que son fundamentales para el monitoreo del clima, la detección de desastres naturales, el seguimiento de cambios en la vegetación y el medio ambiente, y el apoyo a la planificación urbana y agrícola, entre otros usos.

La presencia de satélites en órbita geoestacionaria implica la necesidad de detectar y rastrear con precisión estos objetos espaciales. Esto se debe a que es esencial garantizar un uso eficiente de los recursos orbitales y evitar colisiones con otros satélites, tanto en GEO como en otras órbitas. Además, el monitoreo de los satélites GEO es crucial para mantener la seguridad y la operatividad de los sistemas espaciales y garantizar la continuidad de los servicios de comunicación y observación de la Tierra.

Planteamiento del problema

El desafío "SpotGEO Challenge" o desafío spotGEO plantea un escenario interesante en el campo de la detección de objetos espaciales. En este desafío, se solicita desarrollar un algoritmo capaz de detectar y localizar satélites en órbita geoestacionaria (GEO) a partir de imágenes simples en formato PNG, adquiridas por un telescopio terrestre de bajo costo y de origen desconocido.

La elección de imágenes en formato PNG se debe a su amplia disponibilidad y facilidad de uso en el procesamiento de imágenes. Sin embargo, las imágenes adquiridas por telescopios de bajo costo pueden presentar diversos desafíos. Estos incluyen baja resolución, ruido, distorsiones ópticas y otros artefactos que pueden dificultar la detección precisa de los satélites.

El objetivo principal de este desafío es superar estas dificultades y desarrollar un algoritmo que pueda identificar los objetos en las imágenes provistas. Esto implica analizar y procesar las imágenes para detectar las características específicas de los satélites GEO, como su forma, brillo y posición relativa a otros objetos presentes en la imagen.

Este desafío es relevante porque aborda una problemática práctica en la detección de satélites GEO desde telescopios terrestres de bajo costo. La capacidad de identificar estos satélites de manera precisa y eficiente en imágenes obtenidas con recursos limitados tiene implicaciones importantes en la monitorización espacial, la planificación de comunicaciones y otras aplicaciones relacionadas.

Desafíos y problemas

La detección de satélites en órbita geoestacionaria (GEO) en imágenes adquiridas por telescopios terrestres de bajo costo presenta una serie de desafíos y problemas específicos que deben abordarse en el contexto del desafío spotGEO. Algunos de estos desafíos incluyen:

1. Baja resolución y ruido:
   Las imágenes adquiridas por telescopios de bajo costo pueden tener una resolución limitada, lo que dificulta la identificación precisa de los objetos a detectar. Además, estas imágenes pueden contener ruido y artefactos que afectan la calidad y la claridad de los objetos detectados.

2. Presencia de otros objetos espaciales
   Las imágenes obtenidas por el telescopio pueden contener otros objetos en el espacio, como estrellas, planetas o incluso otros satélites en órbitas distintas a la GEO. Esto provoca que la presencia de estos objetos puede complicar la detección y distinción de los satélites de interés.

3. Variabilidad atmosférica y condiciones de captura
   Las condiciones atmosféricas, como la turbulencia o la presencia de nubes, pueden afectar la calidad de las imágenes capturadas por el telescopio. Estas variabilidades atmosféricas pueden dificultar la detección precisa y afectar la fiabilidad de los resultados.

4. Características específicas de los satélites GEO
   Estos satélites tienen características distintivas, como su forma, brillo y posición relativa en relación con otros objetos espaciales. La detección de estas características específicas puede requerir técnicas avanzadas de procesamiento de imágenes y reconocimiento de patrones para diferenciar correctamente los satélites GEO de otros objetos presentes en la imagen.

Objetivos del trabajo

El presente trabajo tiene como objetivo abordar el desafío spotGEO y desarrollar un algoritmo efectivo para la detección de satélites en órbita geoestacionaria (GEO) en imágenes adquiridas por un telescopio terrestre de bajo costo. Para lograr este objetivo principal, se plantean los siguientes objetivos específicos:

1. Realizar un análisis exhaustivo de la literatura existente
   Se llevará a cabo una revisión detallada de la literatura científica y técnica relacionada con la detección de objetos en imágenes astronómicas, enfoques de procesamiento de imágenes y técnicas de reconocimiento de patrones aplicadas al ámbito espacial.

2. Proponer un algoritmo basado en el conjunto de datos del desafío
   Se desarrollará un algoritmo específico basado en las características de las imágenes proporcionadas en el desafío. El algoritmo deberá ser capaz de detectar y localizar con precisión los satélites GEO en las imágenes ***PNG*** de baja resolución y con ruido, superando los desafíos mencionados anteriormente.

3. Evaluar el rendimiento del algoritmo propuesto
   Se definirán métricas y criterios de evaluación adecuados para medir el rendimiento del algoritmo en términos de precisión, sensibilidad y eficiencia. Se realizarán pruebas exhaustivas del algoritmo utilizando el conjunto de datos proporcionad, y se compararán los resultados obtenidos con los de otros enfoques y algoritmos existentes en el campo.

4. Comparar los resultados con otros enfoques utilizados en el campo
   Se realizará una comparación detallada de los resultados obtenidos con otros enfoques utilizados en la detección de satélites GEO desde telescopios de bajo costo. Esto permitirá evaluar el desempeño y la efectividad del algoritmo propuesto y analizar su potencial en relación con otras técnicas existentes.

Estructura del trabajo

El presente trabajo está estructurado de la siguiente manera para abordar de manera integral el desafío spotGEO y alcanzar los objetivos propuestos:

1. Introducción

   En esta sección introductoria, se presenta el contexto y la relevancia de los satélites en órbita geoestacionaria (GEO), se describe el desafío spotGEO y se plantean los objetivos del trabajo.

2. Estado del arte

   En esta sección, se realiza un análisis exhaustivo de la literatura existente relacionada con la detección de objetos en imágenes astronómicas, enfoques de procesamiento de imágenes y técnicas de reconocimiento de patrones aplicadas al ámbito espacial. Se revisarán y discutirán los enfoques y métodos relevantes utilizados previamente en el campo.

3. Metodología

   Aquí se describe detalladamente la metodología propuesta para el desarrollo del algoritmo de detección de satélites GEO en imágenes del desafío spotGEO. Se explicarán los pasos y las técnicas específicas utilizadas, como el preprocesamiento de imágenes, la extracción de características, la segmentación y la clasificación.

4. Implementación y resultados

   En esta sección, se presentará la implementación del algoritmo propuesto y se mostrarán los resultados obtenidos en la detección de satélites GEO en las imágenes del desafío spotGEO. Se discutirán los resultados y se compararán con otros enfoques existentes en el campo, utilizando métricas apropiadas para evaluar el rendimiento.

5. Discusión y análisis

   En este apartado se llevará a cabo una discusión detallada de los resultados obtenidos y se realizará un análisis crítico de los desafíos, las limitaciones y las posibles mejoras del algoritmo propuesto. Se analizarán las fortalezas y las debilidades del enfoque y se ofrecerán recomendaciones para futuras investigaciones.

6. Conclusiones

   Aquí se resumirán las principales conclusiones del trabajo, destacando los logros alcanzados, las contribuciones realizadas y las implicaciones prácticas de los resultados obtenidos en la detección de satélites GEO desde telescopios terrestres de bajo costo.

7. Referencias bibliográficas

   En este apartado se enumerarán todas las referencias bibliográficas utilizadas en el trabajo, siguiendo un formato de cita adecuado.

Con esta estructura, se busca proporcionar una presentación clara y ordenada de los aspectos fundamentales del trabajo, desde la introducción y la revisión de la literatura hasta la metodología propuesta, la implementación, los resultados, el análisis y las conclusiones. Se pretende que esta estructura permita una comprensión exhaustiva de los enfoques y los resultados obtenidos en la detección de satélites GEO en el desafío spotGEO, así como su relevancia en el ámbito de la observación espacial y la detección de objetos en imágenes astronómicas.

2. Estado del arte

Actualmente, está muy extendido el uso de redes neuronales convolucionales con el fin de detectar objetos en imágenes. Estas redes ya se encuentran presentes en la vida cotidiana de las personas, un ejemplo de ello es el uso de redes convolucionales en coches para la conducción autónoma.

En lo que concierne a este trabajo, a continuación, describiremos las herramientas que existen en el mercado para la detección de objetos en imágenes, como las bibliotecas y ***frameworks*** [1] para el aprendizaje automático, algoritmos de detección de objetos, modelos preentrenados para la detección de objetos, formas de procesar las imágenes y generar mayor cantidad, técnicas de post procesamiento y compañías que ofrecen entrenamiento en la nube.

Bibliotecas y frameworks de Aprendizaje Automático

En el primer punto, discutiremos herramientas y tecnologías relacionadas con el aprendizaje automático que pueden ser útiles para abordar el problema planteado.

- Tensorflow (1): ***TensorFlow*** es una biblioteca de código abierto desarrollada por el equipo de Google Brain que se utiliza principalmente para implementar modelos de aprendizaje automático y realizar operaciones numéricas eficientes en tensores. Los tensores son estructuras matemáticas similares a matrices, y forman la base del procesamiento de datos en ***TensorFlow***.

[1] Marco o esquema de trabajo generalmente utilizado por programadores para realizar desarrollos de programación

Características clave de ***TensorFlow***:

- Abstracción de gráficos computacionales: ***TensorFlow*** representa las operaciones como nodos en un gráfico computacional. Los cálculos se definen mediante grafos, lo que permite optimizar y distribuir eficientemente las operaciones.

- Flexibilidad: Permite definir e implementar diversos tipos de modelos de aprendizaje automático, desde redes neuronales hasta modelos de regresión y clasificación.

- GPU y TPU: Aprovecha el poder de procesamiento de las ***GPU*** y las ***TPU*** (Tensor Processing Units) para acelerar el entrenamiento y la inferencia de modelos.

- Amplia comunidad y soporte: ***TensorFlow*** cuenta con una gran comunidad de usuarios y desarrolladores, lo que facilita el acceso a tutoriales, ejemplos y documentación.

- PyTorch (2): ***PyTorch*** es otro popular ***framework*** de aprendizaje automático de código abierto, desarrollado principalmente por Facebook AI Research (FAIR). Se centra en proporcionar una experiencia más intuitiva y flexible para los desarrolladores, lo que lo hace especialmente popular entre los investigadores y la comunidad académica.

  Características clave de ***PyTorch***:

  - Grafos Computacionales Dinámicos: A diferencia de ***TensorFlow***, ***PyTorch*** utiliza grafos computacionales dinámicos, lo que significa que el grafo se construye sobre la marcha durante la ejecución. Esto facilita el depurado y la experimentación con modelos.

- Fácil de aprender y usar: ***PyTorch*** ofrece una sintaxis más amigable y cercana a ***Python***, lo que simplifica el proceso de definir modelos y realizar tareas de aprendizaje automático.

- Compatibilidad con ***autograd***: ***PyTorch*** proporciona la capacidad de calcular automáticamente gradientes para realizar el aprendizaje basado en retro propagación sin la necesidad de escribir código explícito para calcular gradientes.

- OpenCV (3): ***OpenCV*** es una biblioteca de visión por computadora de código abierto y altamente optimizada. Se utiliza para una amplia variedad de tareas relacionadas con el procesamiento de imágenes y visión por computadora, desde tareas básicas como carga y manipulación de imágenes hasta operaciones más avanzadas como detección de objetos, seguimiento, reconocimiento facial y calibración de cámaras.

  Características clave de ***OpenCV***:

  - Amplia gama de funciones: ***OpenCV*** proporciona una gran cantidad de funciones para procesar imágenes, como filtrado, transformación geométrica, detección de bordes, detección y descripción de características, entre otras.

  - Soporte multiplataforma: ***OpenCV*** es compatible con varios sistemas operativos, lo que facilita su uso en diferentes plataformas.

  - Eficiente y rápido: ***OpenCV*** está altamente optimizado y escrito en C++ y también tiene interfaces para Python y otras lenguas, lo que permite un procesamiento de imágenes rápido y eficiente.

En resumen, ***TensorFlow*** y ***PyTorch*** son ***frameworks*** de aprendizaje automático que permiten construir y entrenar modelos de manera eficiente, mientras que ***OpenCV*** es una biblioteca especializada en visión por computadora para el procesamiento y análisis de imágenes.

Algoritmos de Detección de Objetos

A continuación, se presentan tres algoritmos de detección de objetos en imágenes: ***YOLO*** (You Only Look Once), ***SSD*** (Single Shot Multibox Detector) y ***RetinaNet***. Estos enfoques son conocidos por su eficiencia y rapidez en la detección en tiempo real de objetos en imágenes.

- YOLO (You Only Look Once) (4): ***YOLO*** es un popular algoritmo de detección de objetos en imágenes desarrollado por Joseph Redmon y su equipo en la Universidad de Washington. A diferencia de otros enfoques tradicionales, ***YOLO*** aborda el problema de detección de objetos como un solo problema de regresión, en lugar de dividirlo en varias etapas como detección de regiones y clasificación.

  Características clave de ***YOLO***:

  - Detección en tiempo real: ***YOLO*** es conocido por su rapidez y eficiencia en la detección de objetos en tiempo real debido a su enfoque de regresión única.

  - Predicciones de cuadros delimitadores y clases: ***YOLO*** predice simultáneamente cuadros delimitadores que encierran los objetos detectados y las probabilidades de clase asociadas a esos objetos.

  - Arquitectura basada en redes neuronales convolucionales: ***YOLO*** utiliza una red neuronal convolucional profunda para aprender patrones y características relevantes en las imágenes y realizar predicciones precisas.

- SSD (Single Shot Multibox Detector) (5): ***SSD*** es otro algoritmo de detección de objetos en imágenes que también se enfoca en la eficiencia y velocidad. Fue propuesto por Wei Liu, Dragomir Anguelov, y otros en el Instituto de Tecnología de California (Caltech) y Google Research.

Características clave de ***SSD***:

- Detección en una sola pasada: Similar a ***YOLO***, ***SSD*** realiza la detección de objetos en una sola pasada a través de la red, lo que lo hace más rápido que enfoques que utilizan múltiples etapas.

- Uso de múltiples escalas de características: ***SSD*** utiliza múltiples capas de características con diferentes tamaños para detectar objetos de diferentes escalas en la imagen.

- Predicciones de cuadros delimitadores y clases: Al igual que ***YOLO***, ***SSD*** predice cuadros delimitadores y las probabilidades de clase asociadas a los objetos detectados.

- RetinaNet (6): ***RetinaNet*** es un algoritmo de detección de objetos propuesto por Tsung-Yi Lin, Priya Goyal, Ross Girshick y Kaiming He de Facebook AI Research (FAIR). Se basa en el concepto de enfoque de un solo disparo (single shot), al igual que ***YOLO*** y ***SSD***, pero incorpora una arquitectura de red diferente.

  Características clave de ***RetinaNet***:

  - Focal Loss: ***RetinaNet*** introduce el concepto de ***Focal Loss***, una función de pérdida que aborda el problema del desequilibrio de clases en la detección de objetos. Esto ayuda a mejorar la precisión en la detección de objetos difíciles, que son minoritarios en el conjunto de datos.

  - Uso de anclas: ***RetinaNet*** utiliza anclas, que son regiones predefinidas en la imagen, para generar múltiples propuestas de detección en diferentes escalas y aspectos.

  - Arquitectura de red basada en ***ResNet***: ***RetinaNet*** utiliza una arquitectura de red basada en la red ***ResNet***, que es profunda y ha demostrado su eficacia en tareas de visión por computadora.

Preentrenamiento y Transferencia de Aprendizaje

En este apartado nos enfocaremos en las estrategias de preentrenamiento y transferencia de aprendizaje, dos técnicas fundamentales en el campo del aprendizaje automático que permiten aprovechar modelos previamente entrenados para mejorar el rendimiento y eficiencia de nuevos modelos en tareas específicas, como la detección de objetos en imágenes.

- Preentrenamiento: El preentrenamiento implica entrenar un modelo en un gran conjunto de datos, generalmente utilizando datos no etiquetados, para aprender representaciones significativas y características generales de las imágenes. Estas representaciones se capturan en las capas internas del modelo y reflejan conocimientos previos sobre estructuras, patrones y características presentes en las imágenes.

- Transferencia de aprendizaje (***Transfer Learning***): La transferencia de aprendizaje implica tomar un modelo preentrenado en una tarea relacionada y adaptarlo para resolver una tarea diferente y específica. En lugar de entrenar un modelo desde cero, se inicializan los pesos y parámetros del modelo con los valores aprendidos en el proceso de preentrenamiento. Luego, se ajustan estos pesos mediante un entrenamiento adicional en el nuevo conjunto de datos.

Características clave de Preentrenamiento y Transferencia de Aprendizaje:

- Eficiencia en datos y tiempo: Al preentrenar un modelo en conjuntos de datos masivos, se ahorra tiempo y recursos computacionales, ya que el modelo ha aprendido representaciones generales de imágenes que son útiles para múltiples tareas.

- Mejora del rendimiento: La transferencia de aprendizaje permite mejorar el rendimiento de un modelo en una tarea específica, especialmente cuando se tiene un conjunto de datos pequeño o escaso para el entrenamiento.

- Generalización: Al aprender características generales, los modelos preentrenados y transferidos pueden generalizar mejor a nuevas tareas, incluso con datos limitados.
- Reducción del sobreajuste: La transferencia de aprendizaje ayuda a reducir el riesgo de sobreajuste, ya que el modelo ya ha capturado representaciones relevantes en una tarea más amplia.

Procesamiento de imágenes

Una etapa crucial en el desarrollo de sistemas de detección de objetos en el problema planteado y otras tareas de visión por computadora. A través del procesamiento de imágenes, se aplican diversas técnicas para mejorar la calidad de las imágenes, resaltar características relevantes y preparar los datos para la etapa de detección de objetos.

Características clave del Procesamiento de Imágenes:

- Filtrado y Mejora de Imágenes:
  Se utilizan técnicas de filtrado, como el filtrado gaussiano o el filtrado bilateral, para reducir el ruido en las imágenes y mejorar la calidad visual. La mejora de imágenes ayuda a obtener una representación más clara y nítida de los objetos en la secuencia de imágenes.

- Transformaciones y Aumentación de Datos:
  Se aplican transformaciones geométricas, como rotaciones y cambios de escala, para aumentar la variabilidad del conjunto de datos. La aumentación de datos es especialmente útil cuando se cuenta con un conjunto de imágenes limitado, ya que permite crear versiones modificadas de las imágenes existentes, lo que enriquece el conjunto de entrenamiento.

- Segmentación y Detección de Regiones de Interés:
  La segmentación permite separar regiones de interés, como objetos o áreas de fondo, para enfocar el análisis en las partes relevantes de la imagen. La detección de regiones de interés ayuda a reducir la cantidad de información innecesaria y mejora la eficiencia del proceso de detección de objetos.

- Eliminación de Fondo y Eliminación de Ruido:
  Se aplican técnicas para eliminar el fondo y ruido no deseado en las imágenes, lo que facilita el enfoque en los objetos relevantes. La eliminación de fondo es especialmente útil para resaltar los objetos en el anillo geoestacionario, mientras que la eliminación de ruido mejora la precisión de la detección.

En resumen, el procesamiento de imágenes desempeña un papel fundamental en la preparación de datos para la detección de objetos en el spotGEO Challenge y otras tareas de visión por computadora. La aplicación de técnicas de filtrado, mejora de imágenes, aumentación de datos y eliminación de fondo ayuda a mejorar la calidad, claridad y eficiencia del proceso de detección, lo que contribuye a desarrollar modelos más precisos y efectivos. El procesamiento de imágenes es esencial para resaltar características relevantes, reducir ruido y optimizar el análisis de las secuencias de imágenes capturadas por el telescopio.

Técnicas de posprocesamiento

En este punto se aborda las técnicas de posprocesamiento y otras tareas de detección de objetos en imágenes. Estas técnicas se aplican después de la etapa de detección para mejorar la precisión y consistencia de los resultados obtenidos por el modelo.

Características clave de las Técnicas de Posprocesamiento:

- Fusión de Detecciones:
  La fusión de detecciones es una técnica que combina y agrupa detecciones cercanas que se refieren al mismo objeto en la imagen. Esto ayuda a reducir detecciones duplicadas y mejorar la consistencia del sistema.

- Eliminación de Detecciones Falsas:
  A través de umbrales de confianza y criterios de confiabilidad, se eliminan detecciones con una baja probabilidad de pertenecer a un objeto real. Esto ayuda a reducir falsos positivos y mejorar la precisión general del modelo.

- Posprocesamiento de Bounding Boxes:
  Se pueden aplicar técnicas para ajustar y mejorar las ***bounding boxes*** (cuadros delimitadores) generadas por el modelo de detección. Por ejemplo, el proceso de ***NMS*** (Non-Maximum Suppression) se utiliza para eliminar cuadros delimitadores redundantes o superpuestos, dejando solo las detecciones más relevantes.

- Corrección de Desplazamientos y Escalas:
  En algunas ocasiones, las detecciones pueden tener desplazamientos o escalas incorrectas debido a factores como el movimiento del telescopio o la distorsión de la imagen. Se aplican técnicas para corregir estos desplazamientos y escalas y mejorar la precisión de las detecciones.

- Posprocesamiento de Clasificación:
  Para mejorar la precisión en la clasificación de objetos detectados, se pueden aplicar técnicas de posprocesamiento, como el uso de modelos de lenguaje natural para mejorar la comprensión de los objetos detectados en un contexto más amplio.

Computación en la nube y GPU

Aquí nos enfocaremos en el uso de la computación en la nube y las unidades de procesamiento gráfico (***GPU***) para acelerar el proceso de entrenamiento y la inferencia de modelos de detección de objetos.

Características clave de la Computación en la Nube y ***GPU***:

- Eficiencia en el Entrenamiento de Modelos:
  La computación en la nube ofrece recursos de alto rendimiento, lo que permite acelerar significativamente el entrenamiento de modelos. Los potentes servidores y clústeres en la nube pueden procesar grandes cantidades de datos y cálculos complejos en paralelo, lo que reduce el tiempo necesario para entrenar modelos complejos.

- Paralelización y Procesamiento en Lote:
  Las ***GPU*** son especialmente efectivas para acelerar el procesamiento de datos en lotes. Dado que los modelos de aprendizaje automático realizan operaciones matemáticas intensivas, las ***GPU*** pueden dividir tareas y ejecutar cálculos en paralelo, lo que agiliza el proceso de entrenamiento y predicción.

- Aceleración de Inferencia en Tiempo Real:
  Para aplicaciones en tiempo real, como la detección de objetos en imágenes en tiempo real, la computación en la nube y las ***GPU*** son fundamentales para lograr

inferencia rápida y en tiempo real. Esto permite implementar sistemas de detección en aplicaciones prácticas que requieran una respuesta rápida.

- Escalabilidad y Flexibilidad:
  La computación en la nube ofrece escalabilidad y flexibilidad, lo que significa que puedes ajustar la cantidad de recursos según las necesidades del proyecto. Puedes aumentar o disminuir la cantidad de instancias de ***GPU*** o ***CPU*** en la nube según la complejidad del modelo y la cantidad de datos.

- Reducción de Costos Operativos:
  La computación en la nube ofrece la ventaja de pagar solo por los recursos que se utilizan. Esto puede reducir los costos operativos en comparación con tener que mantener y actualizar una infraestructura de hardware local.

En resumen, la computación en la nube y las ***GPU*** juegan un papel crucial en el desarrollo de sistemas de detección de objetos eficientes y rápidos. Al aprovechar la potencia de cómputo de las ***GPU*** y la escalabilidad de la nube, se pueden acelerar el entrenamiento y la inferencia de modelos, permitiendo implementar soluciones prácticas y precisas. La computación en la nube ofrece flexibilidad y eficiencia en recursos, lo que resulta en un enfoque más rentable para el procesamiento intensivo de datos en tareas de visión por computadora.

3. Metodología

En este apartado, se presenta la metodología utilizada para abordar el desafío de detección de objetos en el spotGEO Challenge. Se describen los pasos clave seguidos para preparar el conjunto de datos, seleccionar y entrenar el modelo de detección, así como las técnicas de posprocesamiento aplicadas.

La metodología empleada se basó en una cuidadosa selección y combinación de técnicas de aprendizaje automático, procesamiento de imágenes y computación de alto rendimiento. El objetivo principal fue desarrollar un sistema de detección efectivo y eficiente, capaz de identificar objetos en el anillo geoestacionario a partir de secuencias de imágenes proporcionadas por un telescopio terrestre de bajo costo.

En primer lugar, se realizó una exhaustiva exploración y análisis del conjunto de datos proporcionado por el spotGEO Challenge. Se examinaron las características de las imágenes, la cantidad de secuencias de imágenes disponibles y los objetos que debían ser detectados en el anillo geoestacionario. Además, se identificaron posibles desafíos asociados con el conjunto de datos, como la baja resolución de las imágenes, la presencia de ruido y distorsiones, así como condiciones adversas de iluminación y atmósfera.

Para preparar los datos para el entrenamiento del modelo de detección, se aplicaron diversas técnicas de procesamiento de imágenes. Se emplearon filtros para reducir el ruido y mejorar la calidad visual de las imágenes. Además, se realizaron mejoras de imagen para obtener una representación más clara y nítida de los objetos en las secuencias de imágenes. Se llevaron a cabo transformaciones geométricas, como rotaciones y cambios de escala, para aumentar la variabilidad del conjunto de datos y mejorar la capacidad de generalización del modelo.

La elección del modelo de detección fue un paso crucial en la metodología. Se evaluaron diferentes arquitecturas de redes neuronales, incluyendo ***YOLO***, ***SSD*** y ***RetinaNet***, considerando la velocidad de detección, la precisión y la eficiencia en el uso de recursos computacionales. Finalmente, se optó por utilizar ***YOLO*** debido a su capacidad para detectar múltiples objetos en una sola pasada sobre la imagen y su rendimiento en tareas similares. La arquitectura ***YOLO*** utiliza una sola red neuronal para realizar tanto la localización como la clasificación de los objetos, lo que lo convierte en una opción prometedora para nuestro escenario de detección de objetos en el anillo geoestacionario.

Con el modelo seleccionado, se procedió a la etapa de entrenamiento. Se configuraron los hiper parámetros, como la tasa de aprendizaje, el tamaño del lote y el número de épocas de entrenamiento, para lograr un equilibrio entre el ajuste del modelo y la prevención del sobreajuste. Durante el entrenamiento, se utilizó una GPU para acelerar los cálculos y reducir significativamente el tiempo necesario para completar el proceso.

Una vez entrenado el modelo, se aplicaron técnicas de posprocesamiento para refinar las detecciones y mejorar la precisión general del sistema. Se aplicaron umbrales de confianza para eliminar detecciones con baja probabilidad de pertenecer a objetos reales, reduciendo los falsos positivos.

La evaluación del modelo se realizó mediante métricas de rendimiento, como la precisión, el ***recall***, el ***F1-score*** y el promedio de precisión media (***mAP***). Se analizaron los resultados obtenidos en el conjunto de prueba y se compararon con otros enfoques existentes y líneas base utilizados en el spotGEO Challenge.

En resumen, la metodología implementada combinó técnicas de procesamiento de imágenes, aprendizaje automático y computación de alto rendimiento para desarrollar un sistema de detección de objetos en el spotGEO Challenge. A través de un enfoque cuidadosamente diseñado y una selección estratégica de herramientas, se logró obtener resultados precisos y eficientes en la detección de objetos en el anillo geoestacionario, abriendo la puerta a futuras aplicaciones en tareas de visión por computadora y detección de objetos en otras áreas de interés.

4. Implementación y resultados

En esta sección, se detalla la implementación del modelo de detección de objetos en el spotGEO Challenge y se presentan los resultados obtenidos mediante la metodología descrita anteriormente. Se describen los pasos específicos seguidos en la implementación y se analizan las detecciones realizadas por el modelo, destacando su precisión y eficacia en el contexto del desafío.

Implementación

- Preparación del Entorno de Desarrollo

  En la fase de preparación del entorno de desarrollo, se tomaron decisiones clave para establecer un ambiente propicio para la implementación del modelo de detección de objetos. Se eligió ***Jupyter Lab*** como el entorno de desarrollo preferido, aprovechando su interactividad y facilidad de uso. Para la implementación del modelo ***YOLO***, se optó por utilizar ***PyTorch***, una biblioteca ampliamente utilizada en tareas de aprendizaje automático.

  La configuración hardware incluyó la ***GPU*** del ordenador, una ***GTX 1060 de 6GB***, para acelerar el proceso de entrenamiento y predicción.

```
import os
import torch

print(torch.cuda.is_available())

os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"
```

Imagen 1: Código en el que se muestra el cambio de valor en la variable de entorno KMP

Para asegurar la compatibilidad con ***YOLO*** y evitar posibles problemas, se ajustó la variable de entorno ***os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"***, se ajustó debido a que al intentar entrenar el modelo, generaba un error relacionado con una librería del tipo ***dll*** la cual ya estaba inicializada, la forma correcta para solucionar este error hubiera sido la creación de un entorno separado en el que instalar todas las dependencias necesarias, se optó por la solución antes mencionada ya que acarreaba menos problemas y era más rápida.

La elección de utilizar un modelo preentrenado, ***YOLO*** en este caso, permitió aprovechar los beneficios de la transferencia de aprendizaje. Para interactuar con ***YOLO*** y realizar inferencias, se instaló el paquete ***ultralytics*** (7), que proporciona una interfaz para utilizar la biblioteca de ***YOLO*** de manera efectiva.

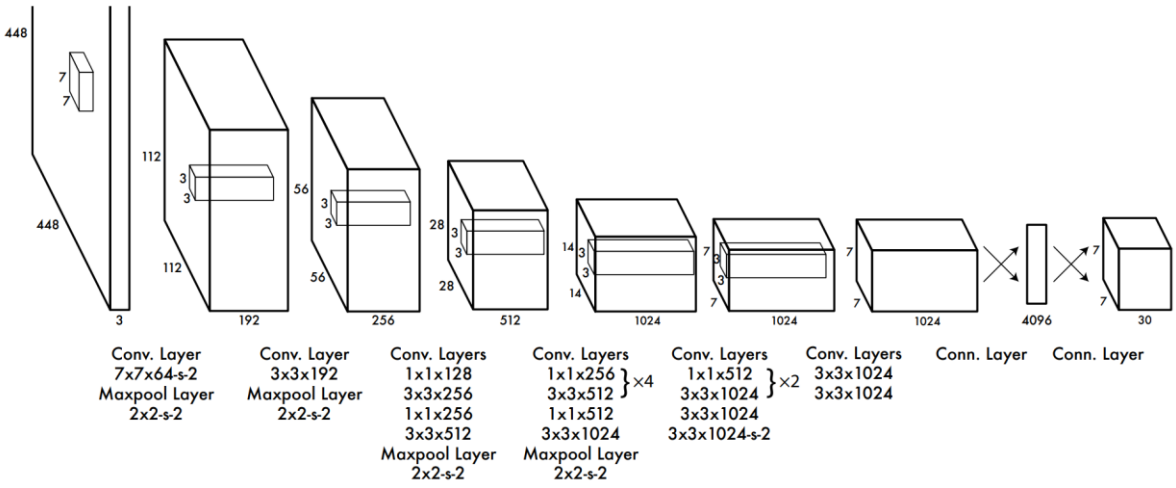

Imagen 2: Arquitectura del modelo YOLO

Durante el desarrollo y ajuste del código, se utilizó la biblioteca ***matplotlib*** para visualizar imágenes generadas en diferentes etapas del proceso. Para cálculos de distancia y otras operaciones matemáticas, la librería ***math*** fue de utilidad. La manipulación de archivos, incluida la apertura y modificación del archivo ***train_anno.json***[2], se llevó a cabo mediante la librería ***json***.

Además, se emplearon las bibliotecas ***os*** y ***cv2*** para manejar aspectos esenciales del desarrollo. La librería ***os*** permitió la creación, eliminación y manipulación de archivos en el sistema, contribuyendo a la gestión de datos. La librería ***cv2*** fue crucial para explorar técnicas de aumento de datos y preprocesamiento de imágenes, mejorando la variabilidad y calidad del conjunto de datos.

Esta cuidadosa configuración del entorno de desarrollo proporcionó una base sólida para la implementación y ajuste del modelo ***YOLO*** permitiendo un enfoque eficiente y efectivo en la tarea de detección de objetos en el anillo geoestacionario.

- Adaptación del Modelo Elegido

En la búsqueda de una solución efectiva para el problema, se exploró la adaptación de un modelo personalizado que combinara capas de convolución (***CNN***), ***TimeDistributed*** y ***Long Short-Term Memory*** (***LSTM***).

[2] Este fichero esta incluido en los datos proporcionados por la ESA y en el se encuentran las coordenadas de todos los objetos en órbita GEO de cada imagen

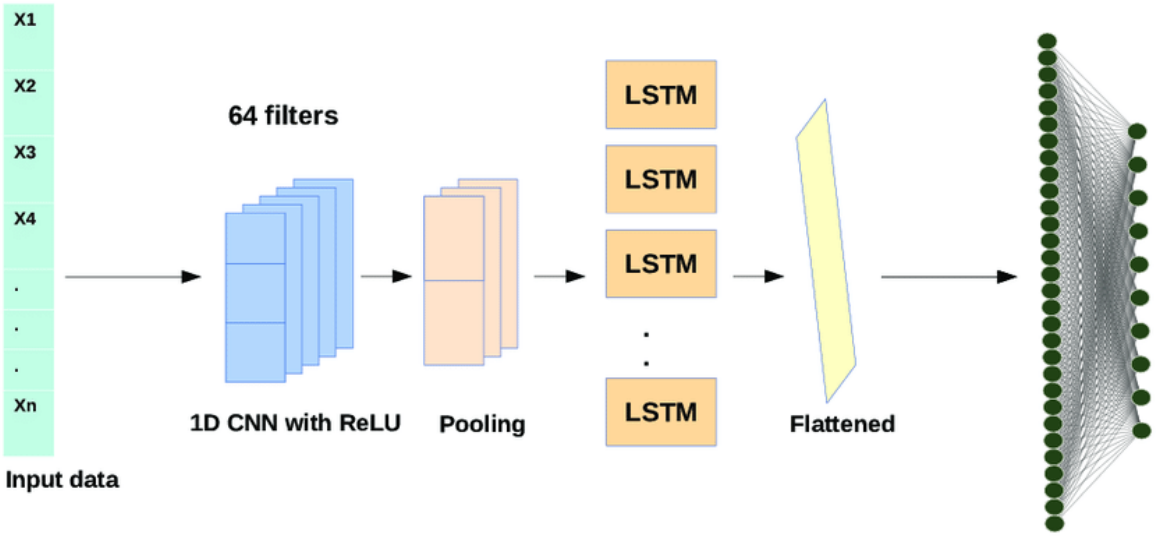

Imagen 3: Arquitectura y distribución del modelo con capas CNN, LSTM

Aunque esta aproximación no fue la opción final seleccionada, su desarrollo y análisis proporcionaron valiosos aprendizajes para la implementación del sistema de detección de objetos.

El modelo personalizado se diseñó siguiendo una estructura multicapa. En primer lugar, se utilizó una instancia de ***InceptionResNetV2*** preentrenada para la extracción de características de las imágenes. Esta red neuronal convolucional profunda es conocida por su capacidad para capturar características complejas en imágenes, lo que lo convirtió en una elección prometedora para esta etapa.

La capa ***TimeDistributed*** se implementó para aplicar la red InceptionResNetV2 a secuencias de cinco imágenes. Esta decisión se basó en la necesidad de considerar la secuencialidad y relaciones temporales presentes en las imágenes del conjunto de datos del spotGEO Challenge. La capa ***Flatten*** se utilizó para transformar las salidas de la capa ***TimeDistributed*** en un formato adecuado para la siguiente capa ***LSTM***.

La capa ***LSTM*** (Long Short-Term Memory) se incorporó para capturar la información secuencial y el contexto temporal en las imágenes. Las capas LSTM son capaces de retener información relevante de eventos anteriores en una secuencia, lo que las hace apropiadas para tratar con datos secuenciales, como las imágenes en secuencias del spotGEO Challenge.

Se añadieron capas ***Dense*** y ***Dropout*** para optimizar y generalizar el modelo. Estas capas contribuyen a la capacidad de aprendizaje y a la reducción del sobreajuste al introducir regularización. La capa final del modelo estaba diseñada para realizar la predicción de detecciones en el formato requerido.

```
inceptionresnet=tf.keras.applications.InceptionResNetV2(
    include_top=False,
    weights="imagenet",
    input_tensor=None,
    input_shape=(height_images,width_images,3)
)

for layer in inceptionresnet.layers[:-4]:
    layer.trainable = False

model = Sequential()

model.add(TimeDistributed(inceptionresnet, input_shape=(5, height_images, width_images, 3)))

model.add(TimeDistributed(Flatten()))

model.add(LSTM(256,activation='relu', return_sequences=False))

model.add(Dense(128,activation='relu'))

model.add(Dropout(.5))
model.add(Dense(5*maximum*2))
model.add(Reshape((5,maximum,2)))

model.compile(optimizer='adam', loss='mean_squared_error', metrics=tf.keras.metrics.RootMeanSquaredError())
model.summary()
```

Imagen 4: Código de la implementación de la arquitectura con capas CNN, LSTM y Time Distributed

A pesar de la prometedora naturaleza de este enfoque personalizado, se decidió no seleccionarlo como la opción final debido a una combinación de factores, como el rendimiento en términos de precisión y eficiencia. Este proceso de adaptación y evaluación fue esencial para el desarrollo iterativo y la mejora continua del sistema de detección de objetos, lo que finalmente condujo a la elección de un enfoque más adecuado y efectivo.

Después de una evaluación exhaustiva y un proceso iterativo, se tomó la decisión de adoptar un enfoque que capitalizara la eficiencia y la capacidad de detección precisa. En ese sentido, se eligió el modelo preentrenado ***YOLOv8n Nano***, una versión compacta de la arquitectura ***YOLO***.

El ***YOLOv8n Nano*** fue especialmente seleccionado porque su cantidad reducida de parámetros (3.2 millones) facilitaba un proceso de entrenamiento más ágil, lo que resultaba esencial dado el tiempo y los recursos disponibles. La capacidad del modelo para procesar imágenes de hasta 640x640 píxeles lo hizo adecuado para el conjunto de datos del spotGEO Challenge.

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.Conv	[128, 128, 3, 2]

Imagen 5: Estructura del modelo YOLO seleccionado

El proceso de entrenamiento se llevó a cabo con un ***batch size*** de 32 debido a la necesidad de limitar la cantidad de ***RAM*** usada y un total de 50 ***épocas***. Se utilizó el optimizador ***Adam***, conocido por su capacidad para adaptar la tasa de aprendizaje durante el entrenamiento.

```
from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(data="D:/TFM/TFM-DATOS/data.yaml", epochs=50, batch=32, optimizer='Adam')
```

Imagen 6: Código de la implementación del modelo YOLO

Durante el proceso de entrenamiento, se observó que la elección de un modelo con menos parámetros podría haber llevado a ciertas limitaciones en términos de la complejidad y la capacidad del modelo para capturar relaciones intrincadas en las secuencias de imágenes. Específicamente, se reconoció que la utilización de modelos con más parámetros podría haber permitido un mayor refinamiento y una adaptación más precisa a las peculiaridades del conjunto de datos. Sin embargo, esta elección también habría requerido una inversión de tiempo y recursos computacionales significativamente mayor.

El proceso de entrenamiento se extendió durante 23 horas continuas, lo que demostró la dedicación y el compromiso en la búsqueda de resultados de alta calidad.

En resumen, el enfoque final de utilizar el modelo preentrenado ***YOLOv8n Nano*** a través de la biblioteca ***PyTorch***, con ***transfer tuning*** y ajuste de parámetros, brindó un sistema de detección eficiente y preciso para el spotGEO Challenge. Aunque se reconoció que modelos con una mayor cantidad de parámetros podrían haber potencialmente mejorado los resultados, la elección pragmática permitió obtener detecciones confiables en el anillo geoestacionario dentro de un marco de tiempo razonable.

- Preparación de Datos (***Preprocesamiento de imágenes***)

En la fase de preparación de dato, se exploraron diferentes enfoques con el objetivo de optimizar y mejorar la calidad de las imágenes antes de introducirlas al modelo de detección. Se describen a continuación los intentos y decisiones tomadas en este proceso:

- División de Imágenes en Cuadrados para Entrenamiento:

  Se experimentó con la división de las imágenes en pequeños cuadrados de 7x7 píxeles, agregando etiquetas para distinguir la presencia de satélites. Sin embargo, se concluyó que esta estrategia no era efectiva, ya que requería dividir cada imagen en numerosas partes, lo que resultaba en una gran cantidad de datos y un tiempo de procesamiento considerablemente largo.

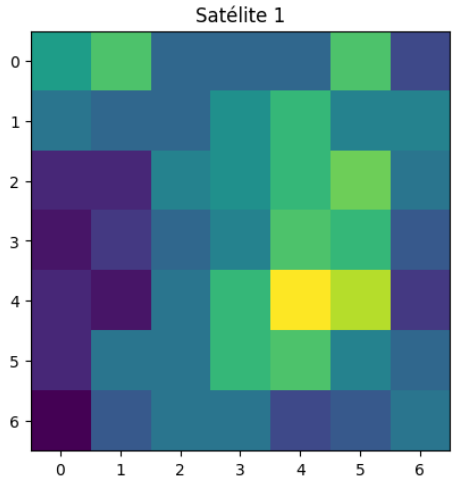

Imagen 7: División de un cuadrado de 7x7 en escala de morados de un satélite

- Detección de Bordes con ***Canny***:

  Se intentó realizar una detección de bordes utilizando el algoritmo ***Canny*** de la librería ***OpenCV***. Sin embargo, surgieron dificultades al intentar eliminar el ruido de fondo sin afectar negativamente la detección de satélites. Los satélites eran a menudo confundidos con el ruido y eliminados incorrectamente, lo que llevó a desechar este enfoque.

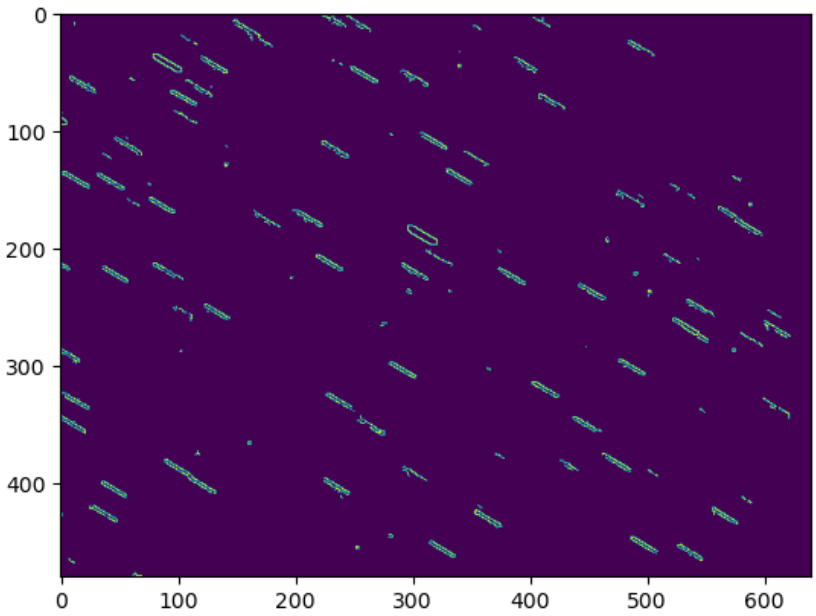

Imagen 8: Ejemplo de imagen donde se implementa el recorte de bordes mediante Canny

- Recorte de Imágenes (***Crop***) y Volteo (***Flip***):

  Se realizó con éxito el recorte de imágenes (***Crop***) como parte de la estrategia de ***data augmentation***. Sin embargo, lo que demostró ser efectivo fue el volteo (***Flip***) tanto vertical como horizontal de las imágenes, lo que contribuyó a diversificar el conjunto de datos y mejorar la capacidad del modelo para reconocer patrones en diferentes orientaciones.

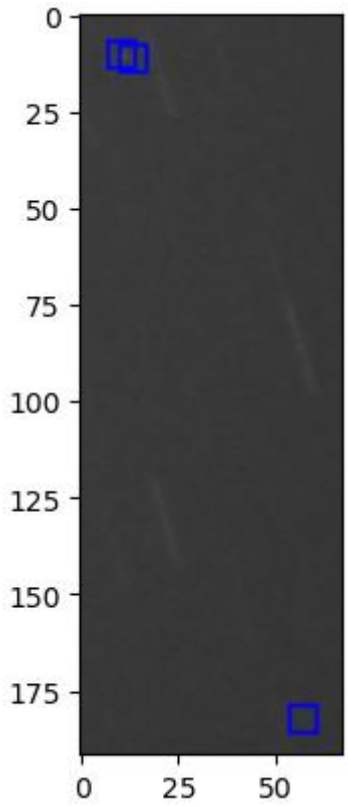

Imagen 9: Recorte (Crop) de una sección de una imágen

- Sistema ***Darknet*** para Utilización de ***YOLO***:

  Para aprovechar el modelo YOLO, se optó por utilizar el sistema ***Darknet***, que implicó la creación de archivos en formato ***Darknet*** para cada imagen. Este proceso fue esencial para que el modelo ***YOLO*** pudiera procesar y realizar inferencias en las imágenes del conjunto de datos.

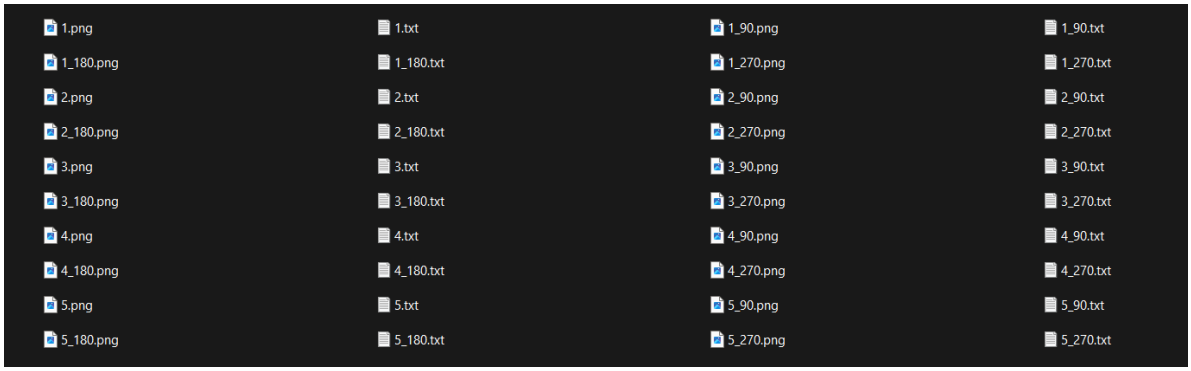

Imagen 10: Ejemplo del preprocesamiento de las imágenes

Como resultado final del preprocesamiento, por cada imagen incluida en los datos aportados para el entrenamiento del modelo se han creado 3 imágenes adicionales, cada una de ellas con su respectivo archivo ***Darknet*** con las coordenadas correctamente modificadas para apuntar a los satélites.

- Posprocesamiento de los resultados

La etapa de posprocesamiento de los resultados se reveló como un componente esencial para filtrar los falsos positivos en la detección de objetos en el spotGEO Challenge. A pesar de la inevitabilidad de descartar algunos positivos genuinos, se abordó este desafío mediante la implementación de un algoritmo que evaluó y conectó los puntos de manera inteligente.

```python
def esta_en_recta(punto1, punto2, punto3, tolerancia=0.01):
    x1, y1 = punto1
    x2, y2 = punto2
    x3, y3 = punto3

    # Calcular pendientes entre los dos puntos conocidos y entre uno de ellos y el tercer punto
    pendiente1 = (y2 - y1) / (x2 - x1) if (x2 - x1) != 0 else float('inf')
    pendiente2 = (y3 - y1) / (x3 - x1) if (x3 - x1) != 0 else float('inf')

    # Verificar si las pendientes son iguales dentro de una tolerancia
    if abs(pendiente1 - pendiente2) < tolerancia:
        x_tend = x1 - x2
        y_tend = y1 - y2

        sigue_x = (x_tend > 0 and (x2 - x3) > 0) or (x_tend < 0 and (x2 - x3) < 0) or (x_tend == 0 and (x2 - x3) == 0)
        sigue_y = (y_tend > 0 and (y2 - y3) > 0) or (y_tend < 0 and (y2 - y3) < 0) or (y_tend == 0 and (y2 - y3) == 0)

        return sigue_x and sigue_y
```

Imagen 11: Código de la implementación del método para calcular la pendiente

En primer lugar, se diseñó un algoritmo que calcula la pendiente entre los puntos de imagen a imagen. Este enfoque permitió identificar y agrupar puntos que forman parte de una misma línea recta, dentro de una tolerancia predefinida. Esta técnica resultó valiosa para diferenciar entre posibles objetos geoestacionarios y ruido aleatorio, ya que los objetos en órbita geoestacionaria se mantienen en posiciones relativamente constantes.

```
def calculate_distance(point1, point2):
    return math.dist(point1[1],point2[1]) / (point1[0] - point2[0])
```

Imagen 12: Código de la implementación del método que calcula la distancia entre dos puntos

En segundo lugar, se aplicó un cálculo de distancia entre los puntos de imagen a imagen y aquellos que formaban una línea recta, priorizando aquellos puntos que estaban más cercanos entre sí. Esta metodología permitió identificar y resaltar aquellas líneas rectas que contenían la mayor cantidad de puntos, lo que a su vez proporcionó una mayor confianza en la detección y confirmación de objetos.

```python
def interpolate_points(start_point, end_point, num_points):
    # Calcula la distancia entre los puntos
    dx = (end_point[0] - start_point[0]) / (num_points + 1)
    dy = (end_point[1] - start_point[1]) / (num_points + 1)

    # Genera los puntos intermedios
    points = []
    for i in range(num_points):
        x = start_point[0] + dx * (i + 1)
        y = start_point[1] + dy * (i + 1)

        points.append((float("{:.2f}".format(x)), float("{:.2f}".format(y))))

    return points

def generate_next_point(prev_point, current_point):
    # Calcula la distancia entre los puntos
    dx = current_point[0] - prev_point[0]
    dy = current_point[1] - prev_point[1]

    # Calcula las coordenadas del siguiente punto
    next_x = current_point[0] + dx
    next_y = current_point[1] + dy

    return [(float("{:.2f}".format(next_x)), float("{:.2f}".format(next_y)))]

def generate_before_point(prev_point, current_point):
    # Calcula la distancia entre los puntos
    dx = prev_point[0] - current_point[0]
    dy = prev_point[1] - current_point[1]

    # Calcula las coordenadas del siguiente punto
    next_x = prev_point[0] + dx
    next_y = prev_point[1] + dy

    return [(float("{:.2f}".format(next_x)), float("{:.2f}".format(next_y)))]
```

Imagen 13: Código de la implementación de los métodos del cálculo de interpolación lineal

Además, se crearon métodos con los que calcular los puntos necesarios para que la línea de coordenadas encontrada pueda señalar todas las coordenadas del objeto en la secuencia, incluso cuando el modelo entrenado no ha sido capaz de encontrar todas ellas. Esto es muy útil cuando la trayectoria de un satélite cruza un objeto brillante como una estrella y el modelo no es capaz de distinguirlo. Este calculo se ha realizado mediante la interpolación lineal.

Se subraya que este enfoque se alinea con la característica única de los objetos geoestacionarios, que permanecen estáticos en el cielo, mientras que el telescopio rota para mantener una posición constante. Esta dinámica dio lugar a la creación de coordenadas interpoladas, un logro adicional que enriqueció la solución. La interpolación de puntos permitió estimar la ubicación potencial del objeto en las imágenes en las que la red neuronal no lo había detectado de manera explícita.

Resultados

- Detecciones Realizadas por el Modelo:

    - Detecciones realizadas correctamente:

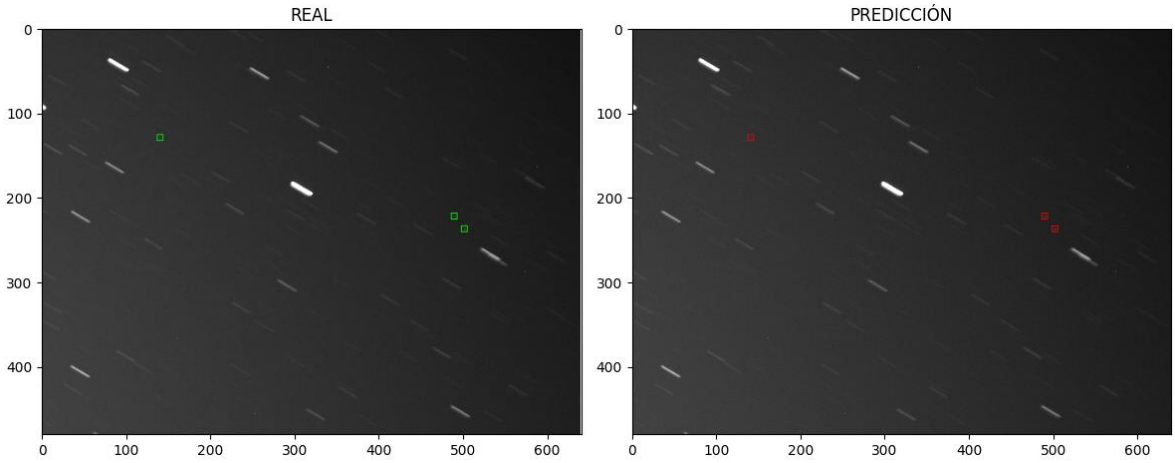

Imagen 14: Secuencia 1, frame 1 (Grupo Train)

Cantidad de satélites predichos: 3

Cantidad de satélites en la imagen: 3

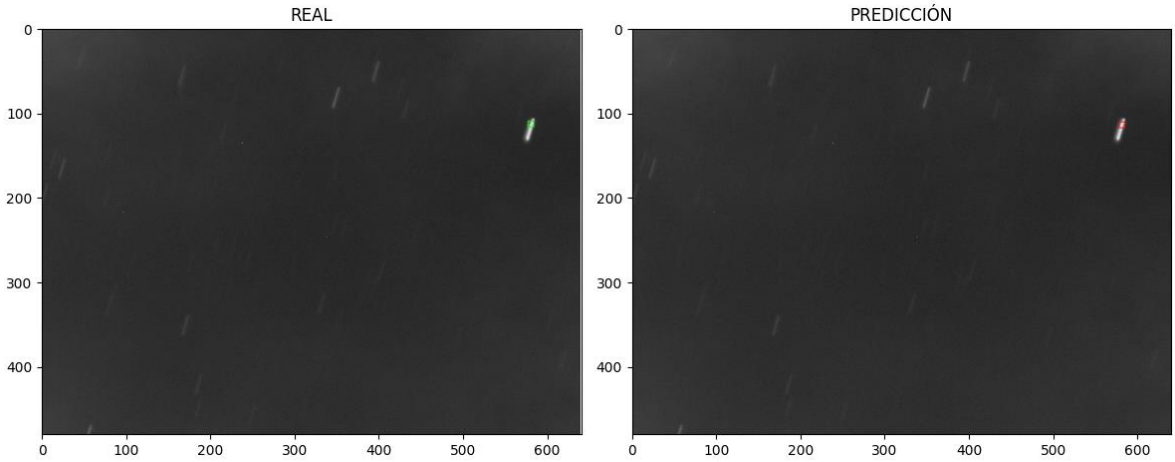

Imagen 15: Secuencia 2, frame 2 (Grupo Train)

Cantidad de satélites predichos: 1

Cantidad de satélites en la imagen: 1

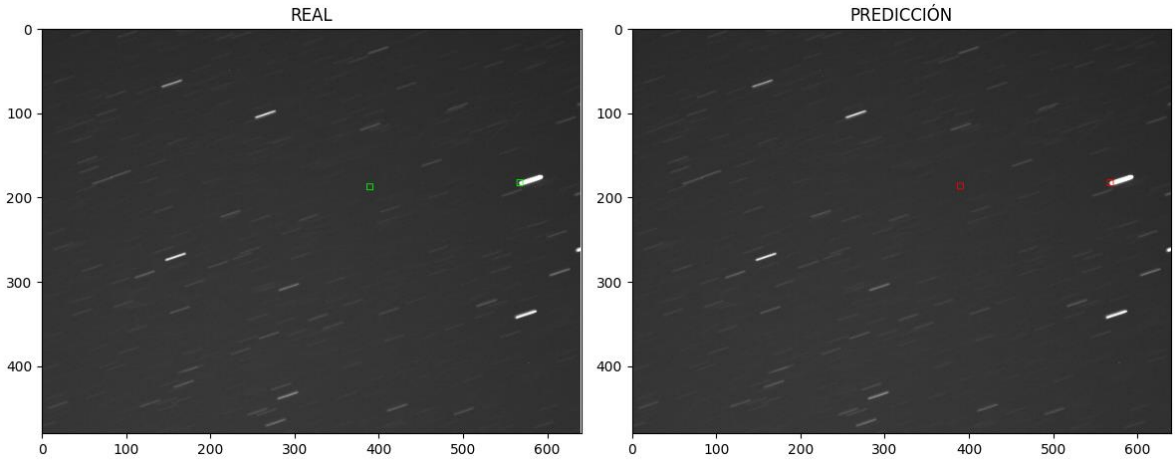

Imagen 16: Secuencia 1, frame 4 (Grupo Test)

Cantidad de satélites predichos: 2

Cantidad de satélites en la imagen: 2

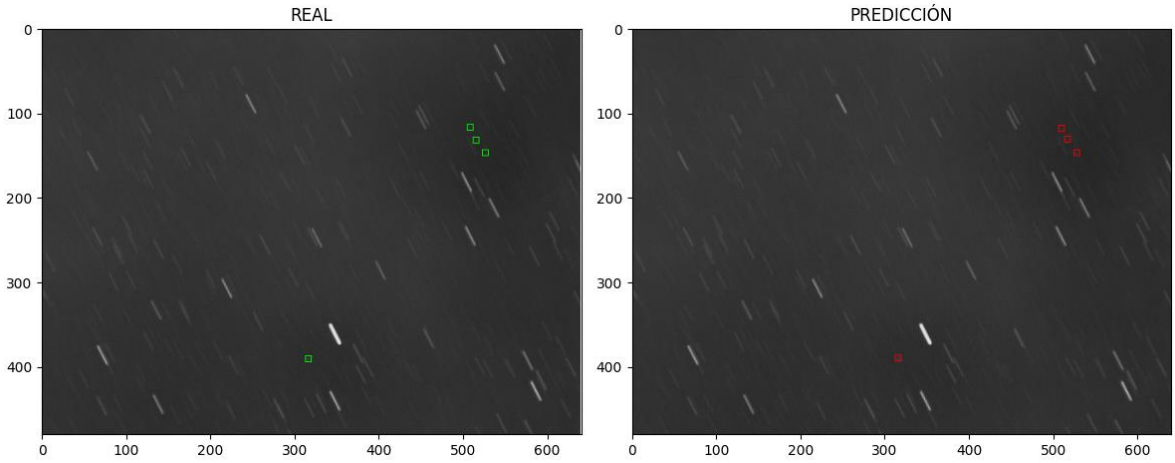

Imagen 17: Secuencia 6, frame 1 (Grupo Test)

Cantidad de satélites predichos: 4

Cantidad de satélites en la imagen: 4

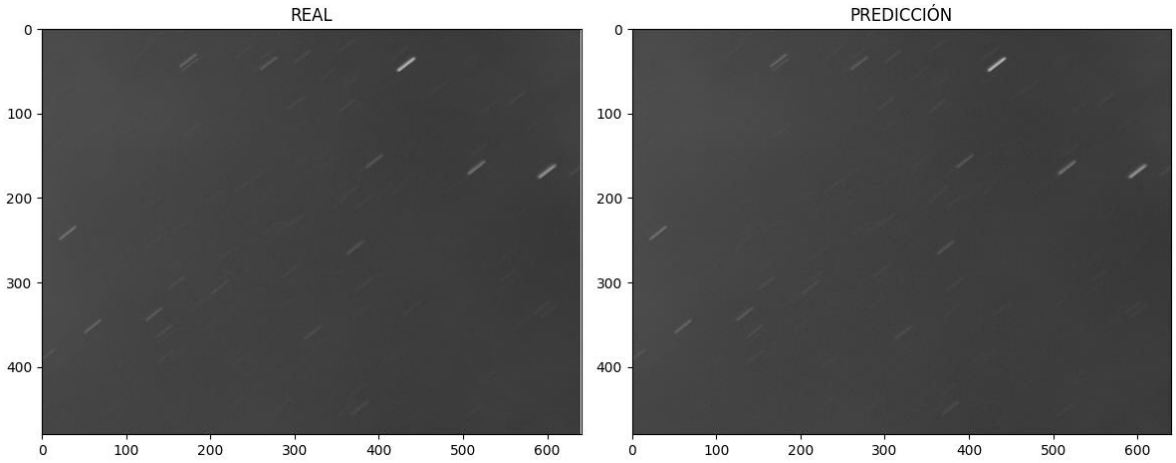

Imagen 18: Secuencia 5, frame 4 (Grupo Test)

Cantidad de satélites predichos: 0

Cantidad de satélites en la imagen: 0

- Detecciones realizadas con errores:

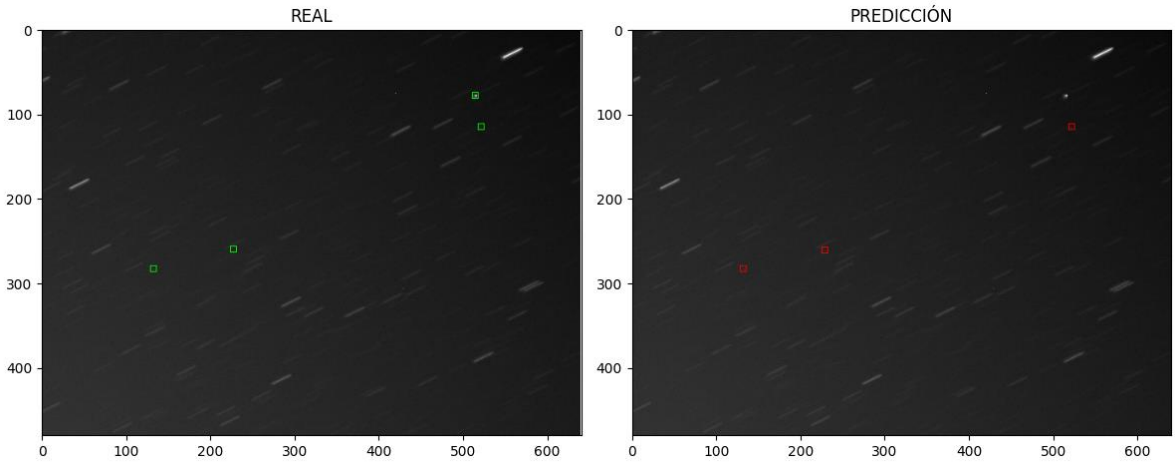

Imagen 19: Secuencia 16, frame 1 (Grupo Train)

Cantidad de satélites predichos: 3

Cantidad de satélites en la imagen: 4

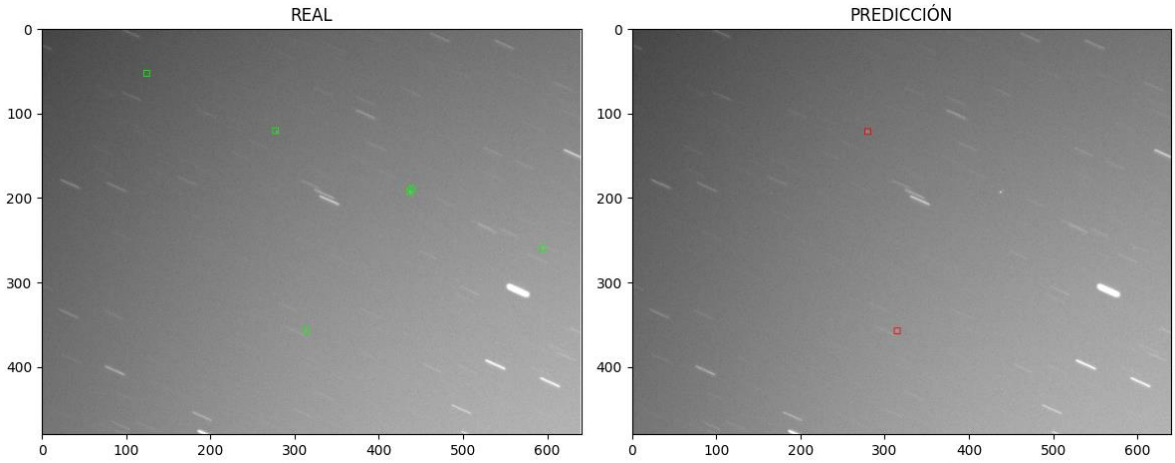

Imagen 20: Secuencia 3, frame 1 (Grupo Test)

Cantidad de satélites predichos: 2

Cantidad de satélites en la imagen: 6

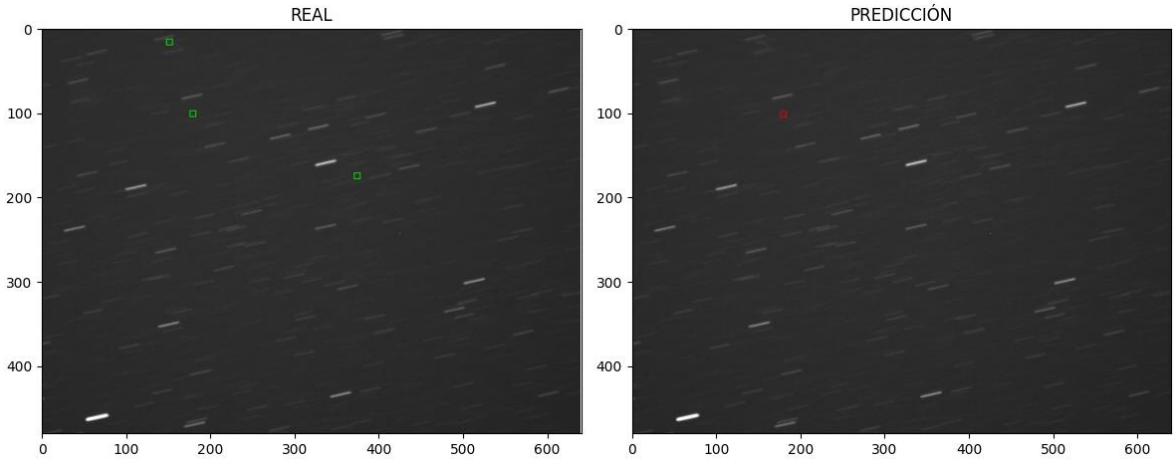

Imagen 21: Secuencia 14, frame 2 (Grupo Test)

Cantidad de satélites predichos: 1

Cantidad de satélites en la imagen: 3

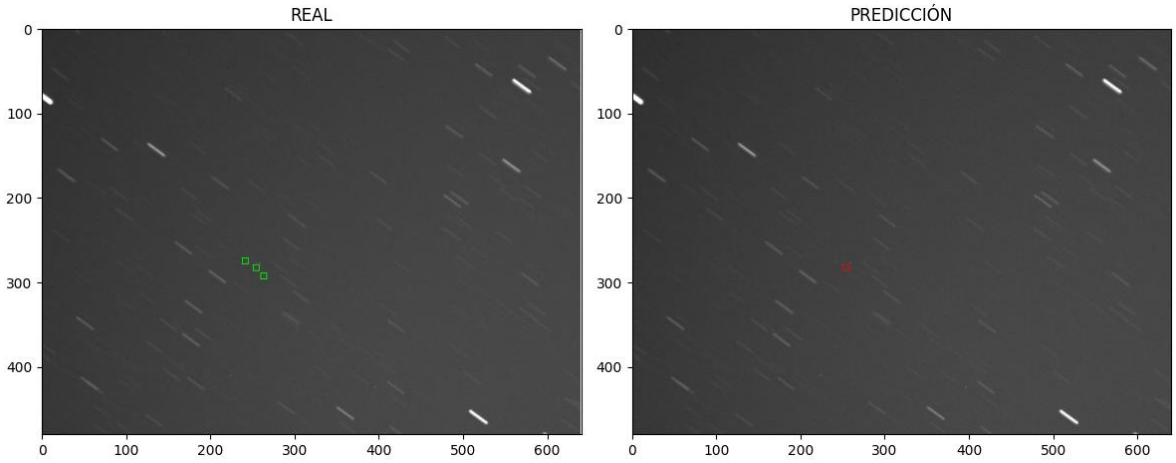

Imagen 22: Secuencia 24, frame 5 (Grupo Test)

Cantidad de satélites predichos: 1

Cantidad de satélites en la imagen: 3

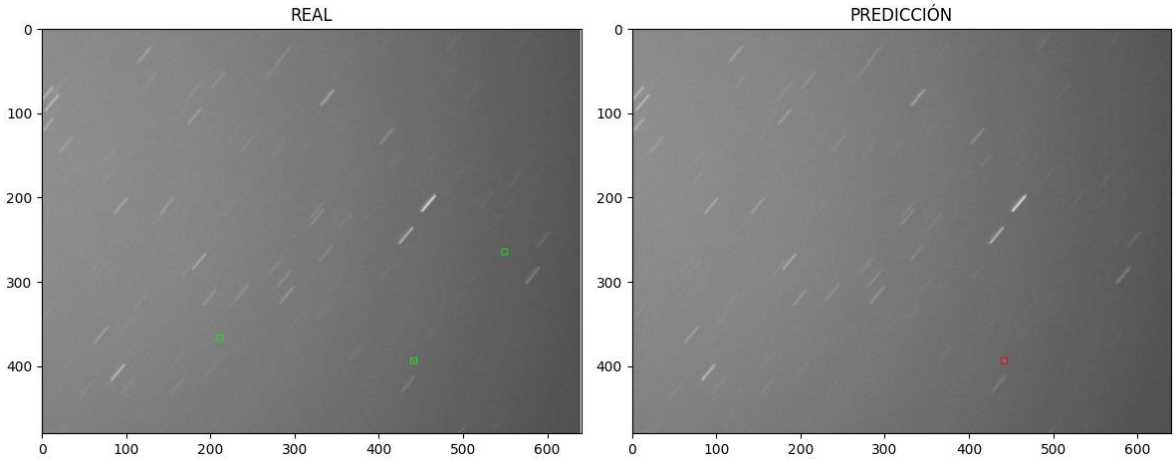

Imagen 23: Secuencia 77, frame 1 (Grupo Test)

Cantidad de satélites predichos: 1

Cantidad de satélites en la imagen: 3

- Métricas de Evaluación:

  - Precisión: La precisión del modelo preentrenado ha sido del 74%.

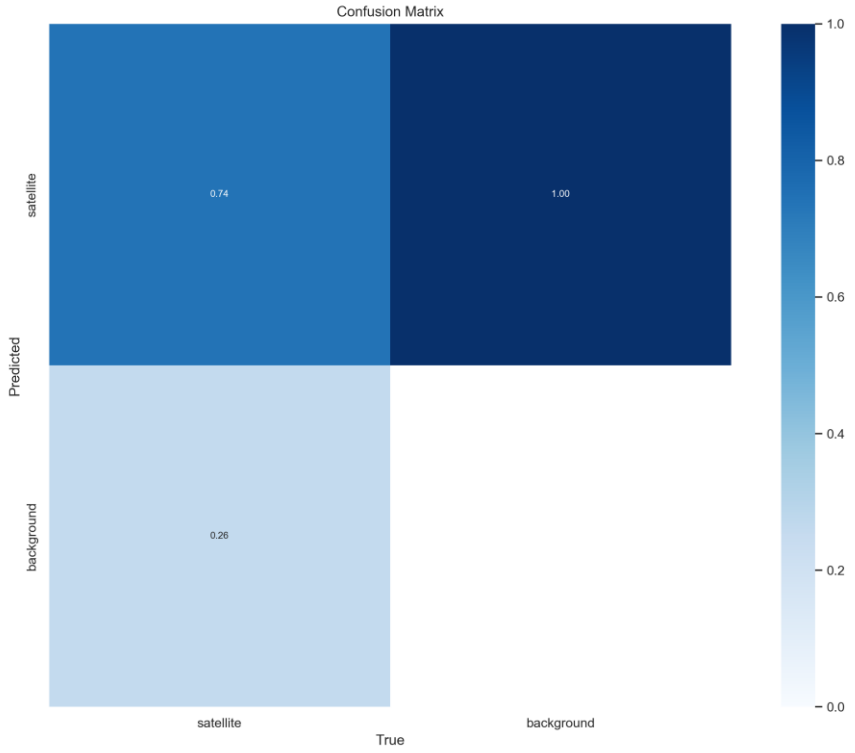

Imagen 24: Matriz de confusión

Estos valores reflejan la capacidad del modelo para identificar correctamente satélites y distinguirlos del fondo en las imágenes. Una puntuación de 0.74 en la celda de satélite predicción y satélite verdad indica un rendimiento bastante positivo en la detección de satélites, mientras que la puntuación de 1.00 en la celda de satélite predicción y fondo verdad indica que hay margen para mejorar la reducción de falsos positivos. Las puntuaciones de 0.26 y 0.00 en las celdas de fondo predicción y satélite verdad, así como fondo predicción y fondo verdad, muestran que hay oportunidades para mejorar tanto en la detección de satélites como en la identificación correcta del fondo.

- Recall:

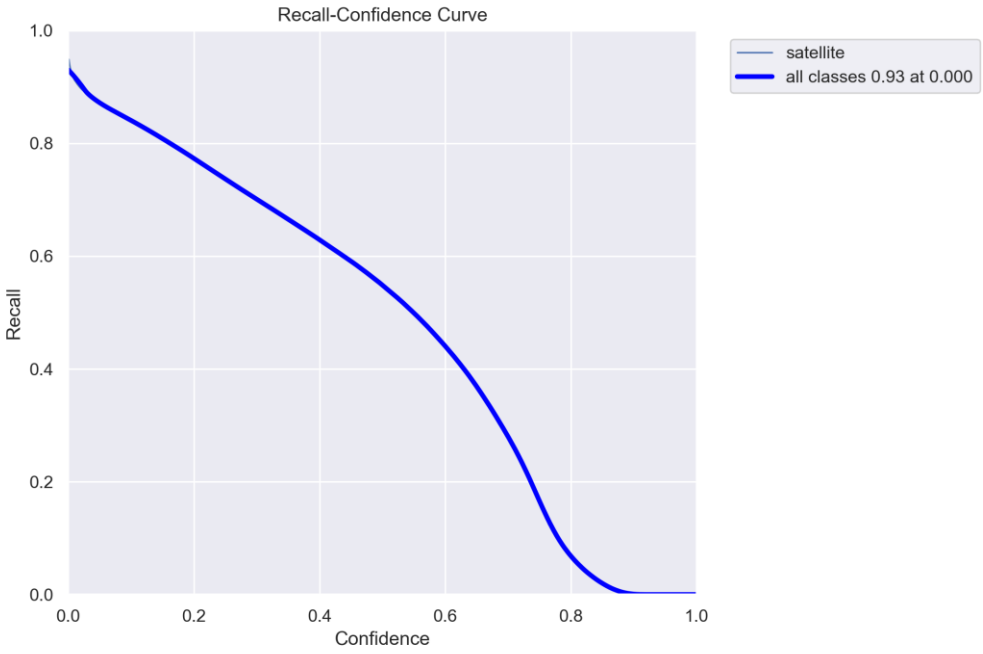

Imagen 25:Curva del Recall del modelo

El ***recall***, que cuantifica la capacidad del modelo para identificar correctamente las instancias positivas, en este caso, los satélites, se presenta en un punto alto de alrededor del 0.92 al comienzo, cuando el umbral de confianza es establecido en su valor mínimo de 0. Esto sugiere que el modelo, al ser más permisivo en sus clasificaciones, logra capturar la mayoría de los satélites presentes en las imágenes, aunque también puede incurrir en un número mayor de falsos positivos debido a la ausencia de requisitos rigurosos de confianza.

Sin embargo, conforme se incrementa gradualmente el umbral de confianza, la curva de ***recall*** experimenta un descenso continuo, indicando que el modelo está adoptando una postura más selectiva y precisa en sus clasificaciones. Esto conlleva a una reducción en la detección de satélites, lo cual es una característica esperada en modelos que persiguen altos estándares de certeza en sus predicciones.

Por lo tanto, determinar el ***recall*** específico del modelo implica considerar el umbral de confianza particular que se emplea. La elección de un umbral concreto influirá en el ***recall*** correspondiente a esa configuración específica. Esta relación entre ***recall*** y umbral de confianza proporciona una visión más matizada y detallada del rendimiento del modelo en diferentes contextos y niveles de confianza, lo que resulta esencial para una evaluación precisa y comprensiva de su capacidad de detección.

- F1-score:

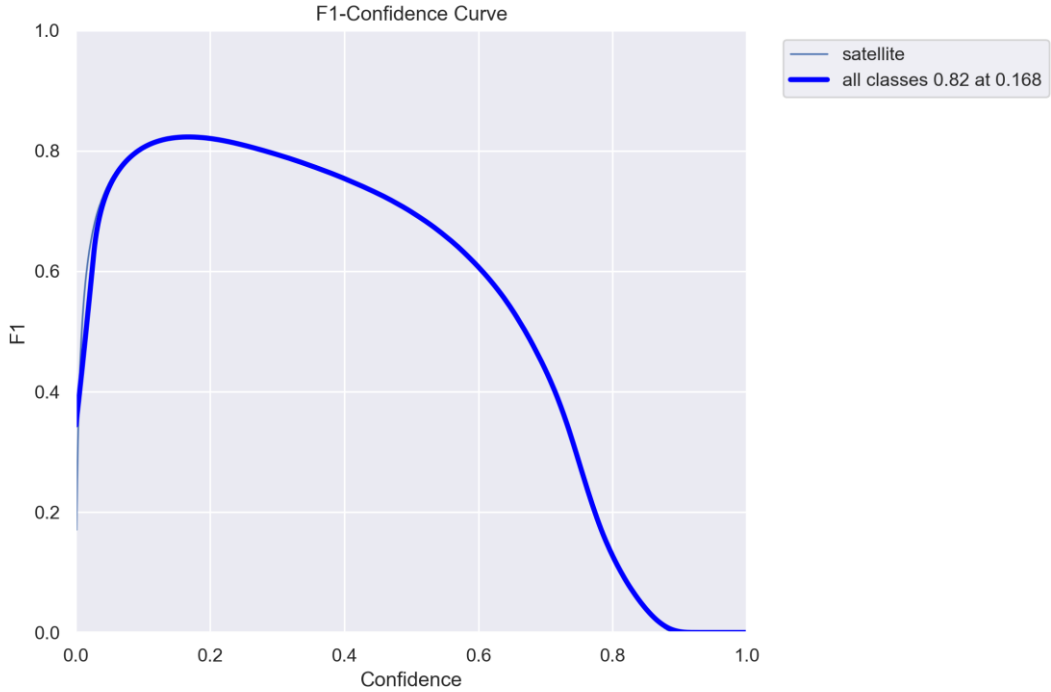

Imagen 26: Curva del F1-score del modelo

La representación gráfica del ***F1-score*** en función de la variación del umbral de confianza presenta una serie de observaciones significativas que proporcionan una comprensión detallada del rendimiento del modelo de detección en diferentes contextos de confianza.

En el extremo izquierdo de la gráfica, cuando el umbral de confianza es establecido en su nivel mínimo de 0, se evidencia un ***F1-score*** inicial de 0.35. Este resultado sugiere que, al no requerir un umbral de confianza para considerar una detección como positiva, el modelo podría estar generando un número considerable de falsos positivos en sus predicciones.

Al aumentar modestamente el umbral de confianza a 0.1, se observa un marcado aumento en el ***F1-score***, alcanzando un valor de alrededor del 0.8. Este aumento indica que el modelo, al aplicar un umbral mínimo de confianza, ha logrado reducir de manera efectiva la inclusión de falsos positivos en sus resultados, obteniendo así un equilibrio más sólido entre la precisión y el ***recall*** en sus clasificaciones.

A partir de este punto, se observa una tendencia gradual de disminución en el F1-score a medida que aumenta el umbral de confianza. Esta disminución gradual sugiere que el modelo está ajustando su enfoque para requerir una confianza más alta en sus predicciones, lo que resulta en una mayor rigurosidad en la aceptación de detecciones. Sin embargo, es importante notar que esta disminución no es lineal, sino más bien un descenso gradual que refleja una adaptación progresiva del modelo.

- Promedio de precisión media (***mAP***):

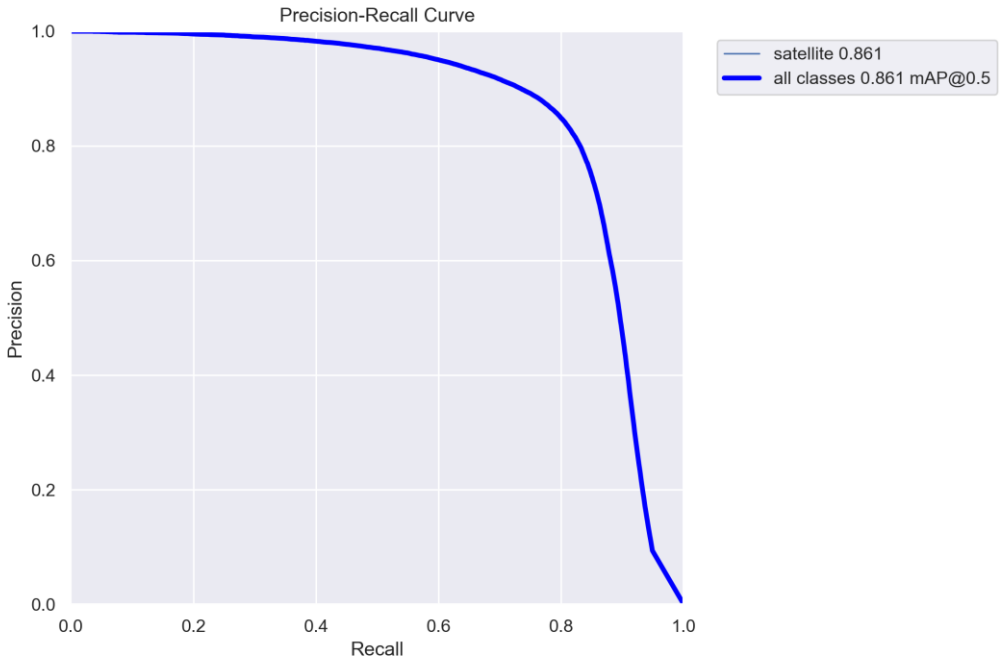

Imagen 27: Curva de la mAP del modelo

La representación gráfica de la métrica ***mAP*** (mean Average Precision) en función de la variación del umbral de confianza revela un patrón de comportamiento que es esencial para comprender el rendimiento integral del modelo de detección de objetos en distintos contextos de confianza.

En el extremo izquierdo de la gráfica, observamos que la precisión comienza en un nivel máximo de 1, acompañado de un ***recall*** de 0. Esto indica que el modelo está aplicando un umbral de confianza excepcionalmente alto, lo que resulta en la detección de un número muy reducido de objetos. Aunque esta configuración ofrece una precisión perfecta, el ***recall*** es mínimo debido a la rigidez del umbral de confianza.

Conforme aumenta el ***recall*** gradualmente, la precisión experimenta una disminución, pero esta disminución es moderada. Esto sugiere que el

modelo está ampliando su margen de detecciones, manteniendo un nivel considerable de precisión en sus resultados. Esta fase de la gráfica refleja una capacidad del modelo para identificar con acierto una mayor proporción de objetos presentes en las imágenes.

Sin embargo, el punto de inflexión se presenta cuando el ***recall*** alcanza 0.8 y la precisión sufre un descenso abrupto a 0.88. Esto implica que, al superar este umbral de confianza específico, el modelo comienza a incluir un número significativo de falsos positivos en sus resultados. Esto puede interpretarse como una situación en la cual el modelo está siendo más permisivo en sus predicciones, pero al hacerlo, está también admitiendo errores en sus detecciones.

Esta dinámica sugiere que existe un punto crítico en el que la precisión se ve comprometida en favor de un mayor ***recall***. Este hallazgo puede estar relacionado con el umbral de confianza establecido, ya que al reducir la confianza requerida para considerar una detección como positiva, el modelo tiende a incluir más detecciones en sus resultados, pero también introduce más falsos positivos.

5. Discusión y análisis

A partir de las métricas discutidas, podemos extraer varias conclusiones valiosas sobre el rendimiento y la efectividad del modelo de detección de objetos en el contexto del problema abordado. Estas conclusiones nos proporcionan una comprensión más profunda de cómo el modelo se comporta en diferentes situaciones y su capacidad para identificar de manera precisa y confiable los objetos de interés en las imágenes.

1. Balance entre Precisión y ***Recall***: La relación inversa entre la precisión y el ***recall*** en las métricas ***mAP*** y ***F1-score*** resalta la importancia de encontrar un equilibrio entre estos dos aspectos. A medida que se ajusta el umbral de confianza, el modelo tiende a favorecer ya sea una mayor precisión (menos falsos positivos) o un mayor ***recall*** (menos falsos negativos). La elección dependerá del contexto específico y de las necesidades del proyecto.

2. Umbral de Confianza Significativo: El punto de inflexión en la gráfica del ***F1-score*** donde la precisión cae abruptamente alrededor de un ***recall*** del 0.8 puede indicar un umbral de confianza crítico. Este umbral puede estar relacionado con la aceptación de falsos positivos. Elegir el umbral adecuado dependerá del grado de certeza requerido y del grado de tolerancia a los errores.

3. Adaptación del Modelo con Confianza: La variación en el ***F1-score*** y la precisión en función del umbral de confianza revela la capacidad del modelo para adaptarse a diferentes niveles de confianza. Esto significa que el modelo puede ser sintonizado para realizar detecciones más conservadoras o más arriesgadas según las necesidades del proyecto y el contexto en el que se aplicará.

4. Efecto en el Rendimiento Global: Estas métricas ofrecen una visión holística del rendimiento general del modelo en una gama de escenarios. Los resultados sugieren que el modelo tiene la capacidad de detectar satélites con cierta precisión, pero el desafío reside en ajustar correctamente los umbrales de confianza para equilibrar la detección precisa con la minimización de falsos positivos.

5. Optimización Futura: Los resultados proporcionan un punto de partida para la optimización continua del modelo. Las observaciones sobre los puntos de inflexión, las relaciones entre ***recall*** y precisión, y los efectos del umbral de confianza ofrecen orientación sobre cómo mejorar la eficacia del modelo a través de ajustes en la configuración.

A partir del análisis del funcionamiento de la red neuronal, su tiempo de entrenamiento, el preprocesamiento y el posprocesamiento de los datos, podemos obtener una visión más completa y enriquecedora de cómo el modelo se adapta y responde a las demandas del problema de detección de objetos en imágenes. Aquí hay algunos análisis que podemos extraer:

1. Eficiencia del Modelo y Tiempo de Entrenamiento: La elección de un modelo preentrenado, en este caso, ***YOLO***, demuestra una estrategia eficiente al aprovechar el conocimiento adquirido en otras tareas de detección de objetos. El hecho de que el modelo ***YOLO Nano*** haya sido seleccionado debido a su menor cantidad de parámetros indica una consideración de eficiencia computacional. El tiempo de entrenamiento de 23 horas continuas para este modelo destaca la complejidad del proceso y la inversión necesaria para lograr resultados de alta calidad.

2. Preprocesamiento de Datos: La exploración de diferentes técnicas de preprocesamiento, como la división de imágenes en pequeños cuadrados y la detección de bordes mediante ***Canny***, revela un enfoque minucioso en la mejora de la calidad de los datos de entrada. La selección de técnicas de aumento de

datos, como el volteo de imágenes, muestra un intento de diversificar el conjunto de entrenamiento para una mejor generalización.

3. Posprocesamiento de Resultados: La implementación de algoritmos para descartar falsos positivos y para unir puntos de detección para formar líneas rectas demuestra un esfuerzo significativo en la refinación de los resultados de la red neuronal. La interpolación de coordenadas para objetos no detectados subraya una búsqueda constante de mejorar y llenar los vacíos en las detecciones.

4. Ajuste de hiper parámetros: La elección de parámetros como el umbral de confianza y la configuración del modelo ***YOLO Nano*** refleja una consideración cuidadosa para lograr el equilibrio entre precisión y ***recall***. El análisis de cómo estos parámetros afectan el rendimiento y la calidad de las detecciones proporciona información valiosa para la optimización futura.

5. Enfoque Iterativo: El análisis de las diferentes estrategias utilizadas, desde el preprocesamiento hasta el posprocesamiento, indica un enfoque iterativo y de mejora continua. El ajuste y experimentación constante demuestra la disposición a adaptarse y aprender de los resultados para lograr un mejor rendimiento.

6. Limitaciones y Desafíos: Los desafíos identificados en el funcionamiento de la red neuronal, como la disminución abrupta de la precisión a ciertos niveles de ***recall*** o la necesidad de combinar técnicas de pre y posprocesamiento para mejorar la calidad de las detecciones, resaltan las limitaciones inherentes y la complejidad del problema.

6. Conclusiones

La profunda exploración realizada en este trabajo arroja luz sobre la implementación de un modelo de detección de objetos en secuencias de imágenes de telescopios de bajo costo, con un enfoque específico en la identificación de satélites en órbita GEO. A través de la selección de herramientas clave como ***PyTorch***, ***YOLO*** y bibliotecas como ***OpenCV*** y ***Ultralytics***, se estableció una sólida base tecnológica para abordar los desafíos únicos de este proyecto.

El corazón de la estrategia adoptada fue la elección de un modelo preentrenado, en este caso, ***YOLO Nano***. Esta elección fue guiada por la necesidad de encontrar un equilibrio entre eficiencia computacional y calidad de detección. La capacidad de adaptar el umbral de confianza permitió sintonizar el modelo para priorizar la precisión o el recall según las necesidades del contexto.

Un aspecto esencial del proceso fue el preprocesamiento y posprocesamiento de los datos. Se exploraron diversas técnicas, como la división de imágenes en cuadrados y la detección de bordes mediante ***Canny***. Si bien algunas técnicas se descartaron debido a la complejidad o al riesgo de confusión con el ruido, otras, como el volteo de imágenes y la interpolación de coordenadas, demostraron ser valiosas para mejorar la calidad de las detecciones.

El análisis exhaustivo de las métricas, como la precisión, el ***recall*** y el ***F1-score***, reveló un equilibrio delicado en la configuración de los umbrales de confianza. La optimización constante y la adaptación de parámetros reflejan un enfoque iterativo para lograr el rendimiento deseado.

En medio de los logros también se presentaron desafíos. La caída abrupta en la precisión a ciertos niveles de ***recall*** y la necesidad de combinar técnicas para refinar los resultados resaltan la complejidad inherente del problema. Estos desafíos también proporcionaron lecciones valiosas para futuros enfoques y mejoras.

En última instancia, las conclusiones extraídas de este análisis resaltan la eficacia de la estrategia adoptada, al tiempo que señalan áreas que pueden beneficiarse de una optimización continua. La combinación de herramientas tecnológicas, la selección de modelos, las técnicas de pre y posprocesamiento y el análisis de resultados forman un enfoque holístico que podría tener un impacto en la detección de objetos en imágenes en aplicaciones futuras.

7. Bibliografía

1. **Google.** TensorFlow. *TensorFlow.* [En línea] https://www.tensorflow.org/?hl=es-419.

2. **Facebook.** PyTorch. *PyTorch.* [En línea] https://pytorch.org/.

3. **Intel.** OpenCV. *OpenCV.* [En línea] https://opencv.org/.

4. **Redmon, Joseph, y otros.** YOLO (You Only Look Once). *YOLO (You Only Look Once).* [En línea] 09 de Mayo de 2016. https://arxiv.org/pdf/1506.02640.pdf.

5. **Liu, Wei, y otros.** SSD: Single Shot MultiBox Detector. *SSD: Single Shot MultiBox Detector.* [En línea] 29 de Diciembre de 2016. https://arxiv.org/pdf/1512.02325.pdf.

6. **Lin, Tsung-Yi, y otros.** Focal Loss for Dense Object Detection. *Focal Loss for Dense Object Detection.* [En línea] 07 de Febrero de 2018. https://arxiv.org/pdf/1708.02002.pdf.

7. **glenn-jocher, y otros.** Ultralytics YOLOv8. *Ultralytics YOLOv8.* [En línea] 05 de Diciembre de 2022. https://docs.ultralytics.com/.

8. **Europea, Agencia Espacial.** Kelvins - ESA's Advanced Concepts Competition Website. *Kelvins - ESA's Advanced Concepts Competition Website.* [En línea] 8 de Junio de 2020. https://kelvins.esa.int/spot-the-geo-satellites/.

9. **Fitzmaurice, Joshua, y otros.** Detection and correlation of geosynchronous objects in NASA’s Wide-field Infrared Survey Explorer images. *Detection and correlation of geosynchronous objects in NASA’s Wide-field Infrared Survey Explorer images.* [En línea] Junio de 2021. https://www.sciencedirect.com/science/article/abs/pii/S0094576521001211.

10. **Chen, Bo, y otros.** Spot the GEO Satellites: From Dataset to Kelvins SpotGEO Challenge. *Spot the GEO Satellites: From Dataset to Kelvins SpotGEO Challenge.* [En línea] https://openaccess.thecvf.com/content/CVPR2021W/AI4Space/html/Chen_Spot_the_GEO_Satellites_From_Dataset_to_Kelvins_SpotGEO_Challenge_CVPRW_2021_paper.html.

11. **Prakash, Soorya.** Hands-On Practice with Time Distributed Layers using Tensorflow. *Hands-On Practice with Time Distributed Layers using Tensorflow.* [En línea] https://levelup.gitconnected.com/hands-on-practice-with-time-distributed-layers-using-tensorflow-c776a5d78e7e.

12. **Dai, Yuqi, y otros.** Effective Multi-Frame Optical Detection Algorithm for GEO Space Objects. *Effective Multi-Frame Optical Detection Algorithm for GEO Space Objects.* [En línea] 06 de Abril de 2022. https://www.mdpi.com/2076-3417/12/9/4610/htm.