

SISTEMA DE VISIÓN COMPUTACIONAL DISTRIBUIDO PARA DETECCIÓN DE OBJETOS

OBJETIVO GENERAL:

Desarrollar un sistema distribuido de visión computacional para la detección de objetos en imágenes, optimizando la escalabilidad y la eficiencia del procesamiento.

OBJETIVOS ESPECÍFICOS:

SPRINT 1

- Configurar el entorno de desarrollo con las bibliotecas necesarias.
- Recopilar y preprocesar un conjunto de datos de imágenes.
- Implementar una red neuronal simple para la detección de objetos en un entorno local.

SPRINT 2

- Distribuir el procesamiento de imágenes utilizando técnicas de paralelismo.
- Implementar una cola de tareas para gestionar el procesamiento de imágenes.
- Optimizar la red neuronal para ejecución distribuida.

SPRINT 3

- Desarrollar una API REST para el procesamiento de imágenes.
- Implementar un frontend básico para subir y visualizar imágenes procesadas
- Preparar y realizar una presentación del sistema.

Metodología

- **Metodología ágil:** La metodología ágil permitió el desarrollo incremental en cada sprint, lo cual impactó en la reducción del tiempo de entrenamiento de la red neuronal al implementar técnicas de paralelismo.

- **Estructura de los sprints:**

- Sprint 1:

- Tareas planificadas:
 - Tarea 1: Configurar el entorno de desarrollo
 - Tarea 2: Recopilar y preprocesar un conjunto de datos de imágenes:
 - Recopilación de imágenes
 - Preprocesamiento del conjunto de imágenes
 - Tarea 3: Implementar una red neuronal simple para la detección de objetos en un entorno local:
 - Creación de la arquitectura de la red neuronal

- Sprint 2:

- Tareas planificadas:
 - Tarea 1: Distribuir el procesamiento de imágenes utilizando técnicas de paralelismo:
 - Pool de procesos para la data de train y test
 - Tarea 2: Uso de GPU para mejora del tiempo en el entrenamiento de la red neuronal
 - Tarea 3: Cuantización post-entrenamiento de la red

- **Herramientas y tecnologías utilizadas:**

- **Numpy:** Manejo de datos
- **xml:** Leer los archivos xml
- **os:** Obtener las rutas de los archivos
- **PIL:** Manipulación de imágenes
- **Tensorflow & Keras:** Creación de redes neuronales, ejecución distribuida en varias unidades de procesamiento y cuantización de modelos.
- **concurrent.futures:** Distribución del procesamiento de imágenes a cada proceso

Objetivos y logros del Sprint 1

OBJETIVOS

- Configurar el entorno de desarrollo con las bibliotecas necesarias.
- Recopilar y preprocessar un conjunto de datos de imágenes.
- Implementar una red neuronal simple para la detección de objetos en un entorno local.



LOGROS

- Las funciones para el preprocessamiento de los datos cumplían el rol que se quiso lograr en un principio.
- Si bien la red no tuvo una buena precisión, se logró crear una arquitectura de red neuronal para los dos problemas en la detección de objetos: clasificación para identificar el objeto que se encuentra en la imagen, así como el problema de regresión para que la red pueda predecir los valores de los bounding boxes para el objeto presente en la imagen.

Objetivos y logros del Sprint 2

OBJETIVOS

- Distribuir el procesamiento de imágenes utilizando técnicas de paralelismo.
- Implementar una cola de tareas para gestionar el procesamiento de imágenes.
- Optimizar la red neuronal para ejecución distribuida.



LOGROS

- Se pudo asignar una tarea a cada proceso, funcionando así el paralelismo a nivel de datos.
- El tiempo de entrenamiento usando GPU fue mejorado.
- El espacio que ocupa en memoria el modelo se logró, al aplicar la cuantización post-training al modelo.

RESULTADOS

FUNCIONALIDADES DESARROLLADAS

SPRINT 1

- Tareas implementadas:
- Configuración del entorno
- Preprocesamiento de imágenes
- Creación de la arquitectura de la red neuronal

SPRINT 2

- Uso de "concurrent futures" para distribuir el procesamiento de los imágenes utilizando técnicas de paralelismo
- Uso de GPU para el entrenamiento
- Cuantización post-entrenamiento

RESULTADOS DE PRUEBAS Y ANÁLISIS DE RENDIMIENTO

SPRINT 1

PRUEBAS:

- Prueba 1: Se entrenó al modelo con 10 épocas sin considerar GPU.
- Se utilizaron 4 imágenes de prueba para evaluar los valores predichos por el modelo.
- Accuracy para el problema de clasificación: 13.1944477558136 %
- MSE para el problema de bounding boxes: 1816.4044189453125

- **Prueba 1: 10 épocas sin GPU:**

- Tiempo de entrenamiento: 12 minutos para 10 épocas

Prueba 2: Se entrenó al modelo con 20 épocas sin considerar GPU.

- Se utilizaron 4 imágenes de prueba para evaluar los valores predichos por el modelo.
- Accuracy para el problema de clasificación: 16.66666716337204 %
- MSE para el problema de bounding boxes: 1560.8096923828125

- **Prueba 2: 20 épocas sin GPU:**

- Tiempo de entrenamiento: 24 minutos para 20 épocas

RESULTADOS

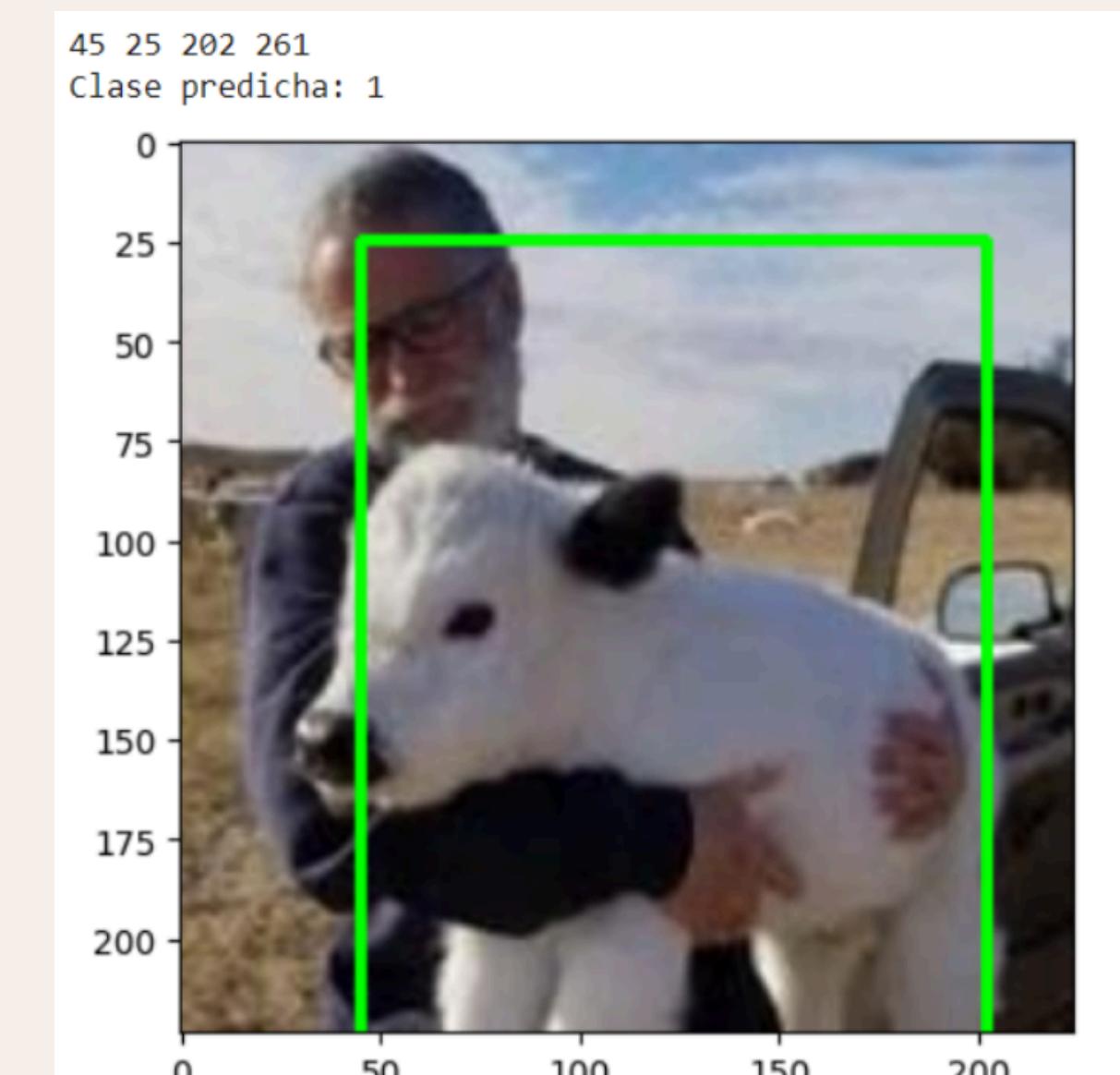
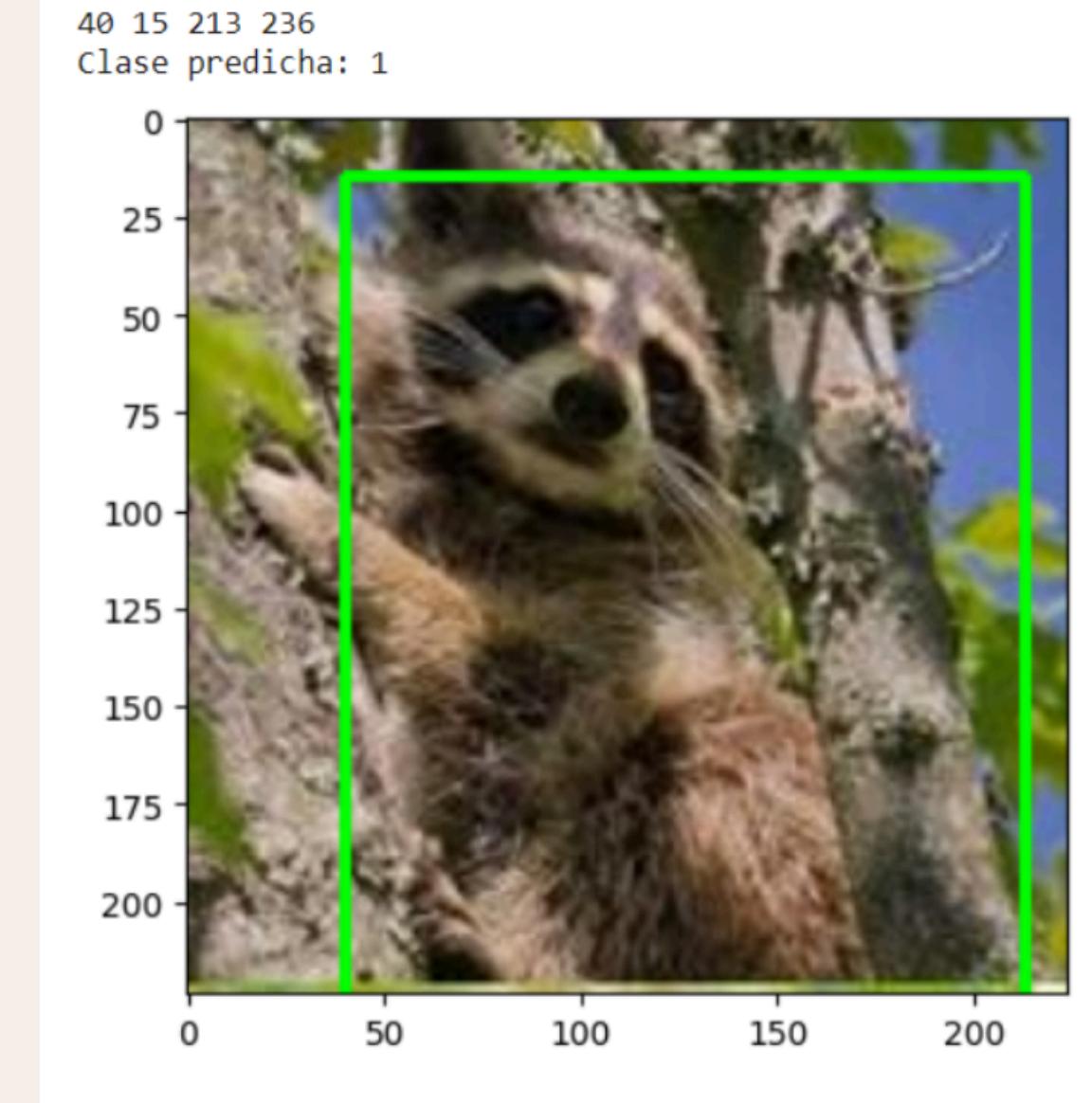
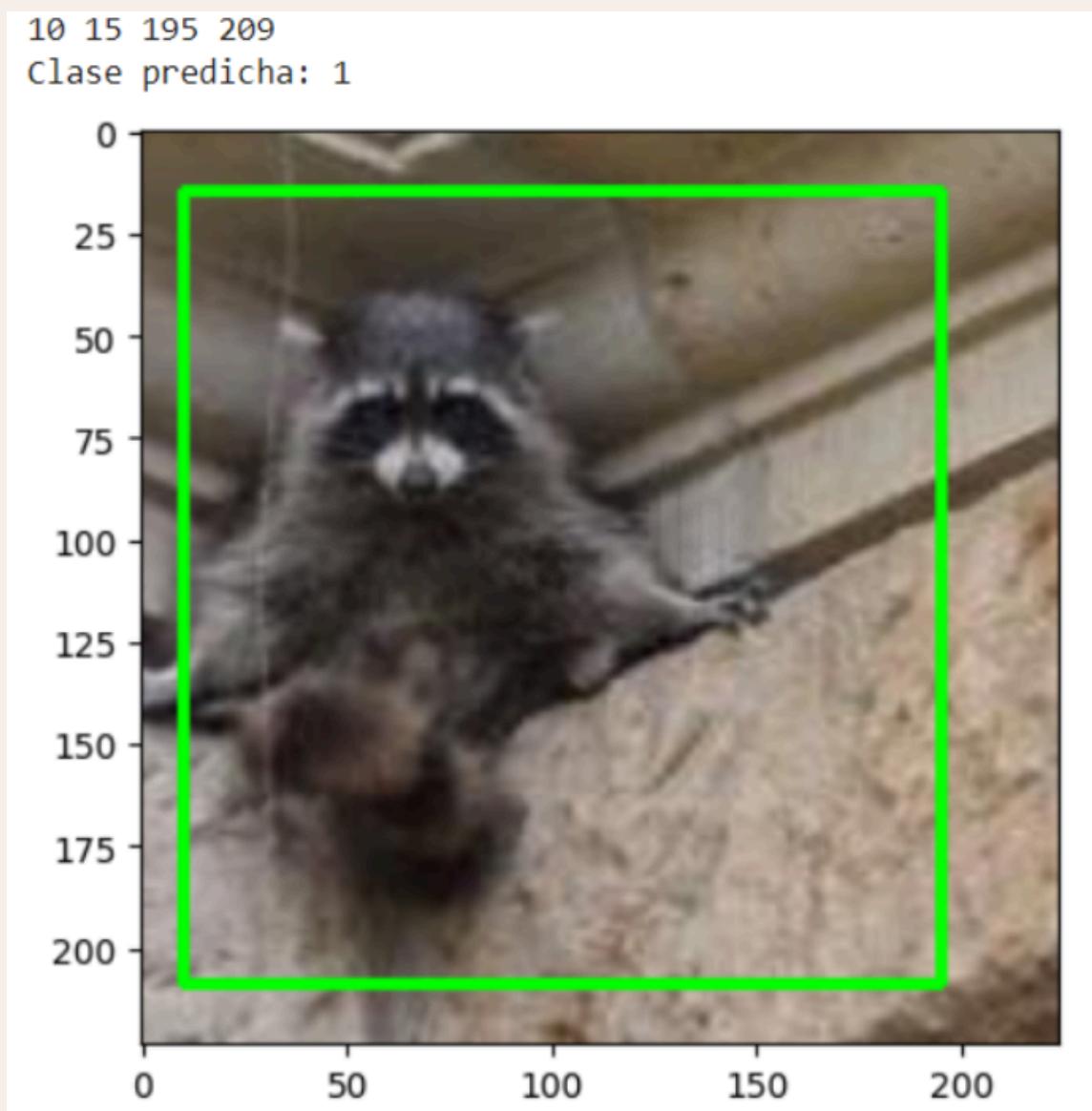
RESULTADOS DE PRUEBAS Y ANÁLISIS DE RENDIMIENTO

SPRINT 2

- Prueba 1: Se entrenó al modelo con 10 épocas considerando GPU.
 - Se utilizaron 4 imágenes de prueba para evaluar los valores predichos por el modelo.
 - Accuracy para el problema de clasificación: 16.6 %
 - MSE para el problema de bounding boxes: 1753.95
- 1) Entrenamiento de la red con GPU y con 10 épocas:
- Tiempo de entrenamiento: 47 segundos para 10 épocas
- Prueba 2: Se entrenó al modelo con 20 épocas considerando GPU.
- Se utilizaron 4 imágenes de prueba para evaluar los valores predichos por el modelo.
- Accuracy para el problema de clasificación: 16.6 %
- MSE para el problema de bounding boxes: 1588.02
- 2) Entrenamiento de la red con GPU y con 20 épocas:
- Tiempo de entrenamiento: 82.47 segundos (1 minuto con 22 segundos) para 20 épocas

DEMOSTRACIÓN

SPRINT 1



[LINK DE VIDEOS DEL SPRINT 1](#)

DEMOSTRACIÓN

SPRINT 2

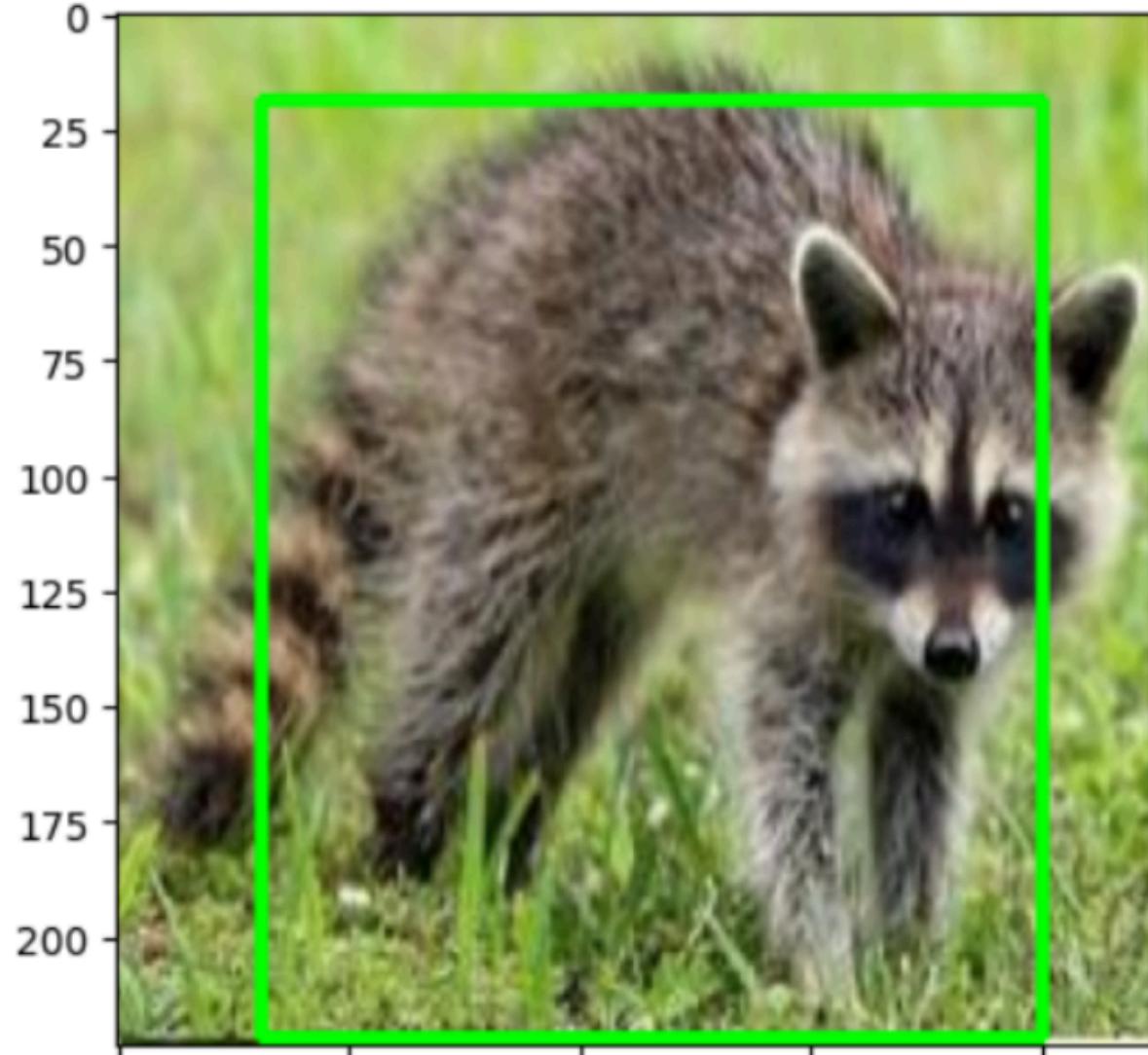
31

19

200

222

Clase predicha: 7



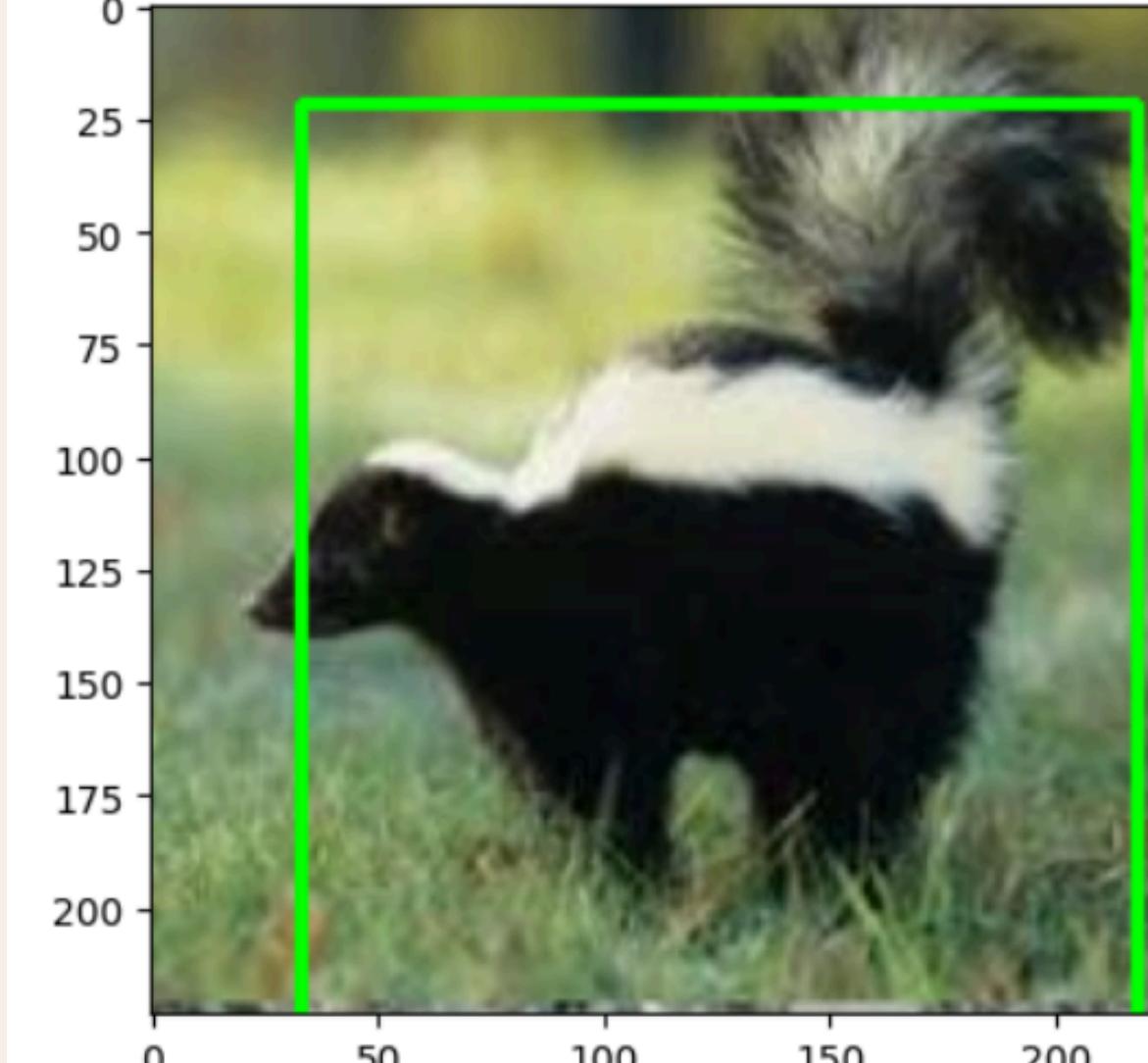
33

22

218

227

Clase predicha: 7



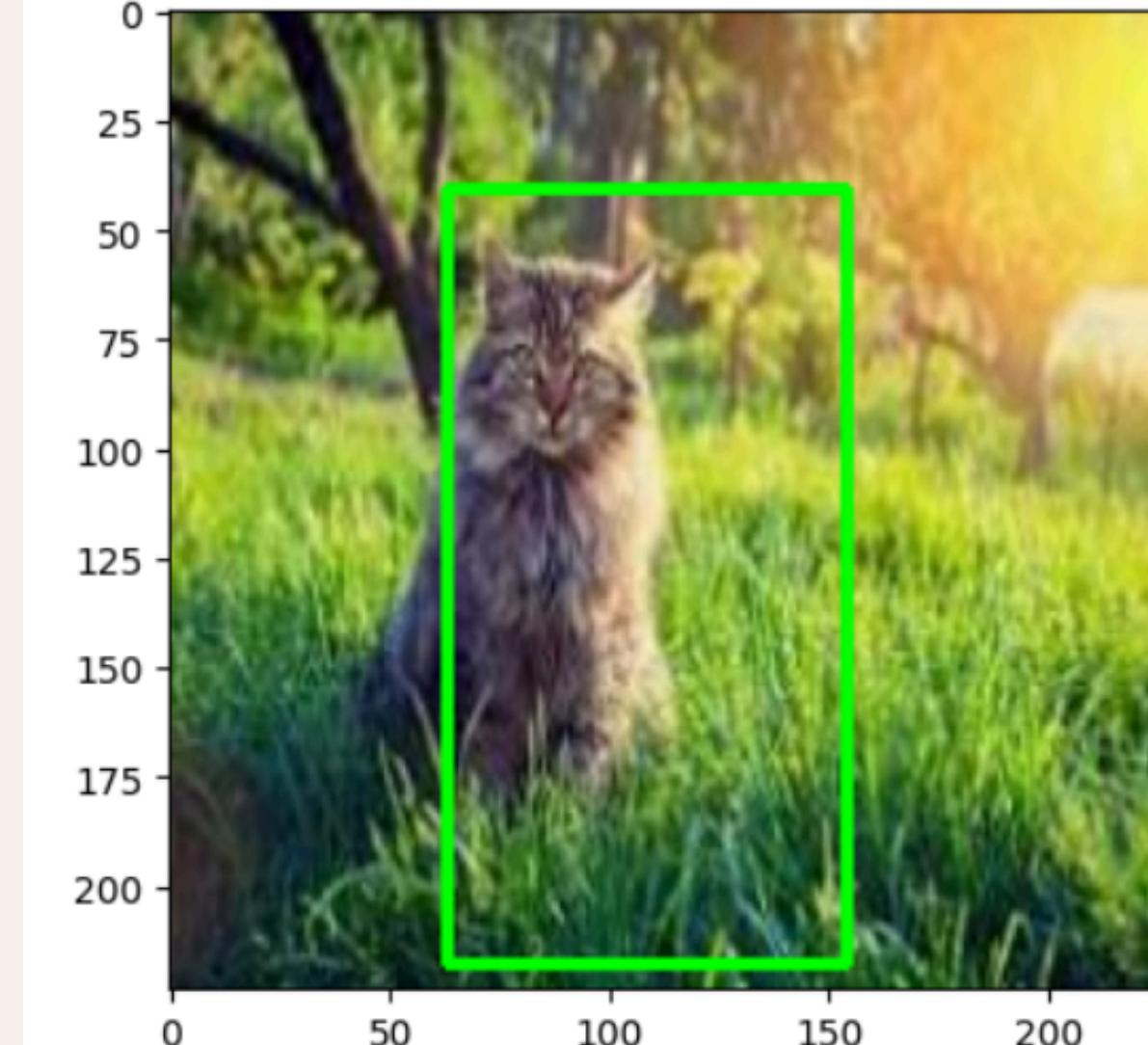
63

41

154

218

Clase predicha: 7



[LINK DE VIDEOS DEL SPRINT 2](#)

Análisis - evaluación

Lecciones aprendidas

- El procesamiento de las imágenes juega un papel crucial para el entrenamiento de la red neuronal. Por lo que una mayor variedad de las mismas son fundamentales para que la red se entrene de mejor manera y generalice bien.
- La arquitectura de la red neuronal, que incluye las capas de convolución y MaxPooling, pueden determinar qué tan bien la red puede captar características (extracción de características) relevantes después de cada filtro o convolución aplicado.
- El uso de GPU es crucial para el entrenamiento de la red. Esta genera mayor rapidez durante el entrenamiento de la red neuronal; sin embargo, si solo se usa CPU, el proceso de entrenamiento tardará más, como fue el caso en este sprint 1.
- Se pudo usar el paralelismo a nivel de datos, aplicando la misma instrucción (los pasos para el procesamiento de las imágenes) a un conjunto de datos, repartiéndolo a cada proceso.
- El uso de GPU puede acelerar exponencialmente si se compara a entrenar la red con una CPU. -La cuantización de modelos; en este caso, cuantización post-training, puede optimizar el uso de la memoria.



Desafíos

- La red no tuvo una buena precisión para la tarea de clasificación. En lo que respecta al tiempo de entrenamiento que la red se demoró, se pudo mejorar haciendo uso de la GPU, la unidad de procesamiento ideal para manejar las operaciones matriciales y vectoriales, fundamentales en la etapa de training para ajustar los pesos de la red neuronal. En el siguiente sprint se hará uso del GPU y se comparará el tiempo tomado.
- Respecto a la cola de tareas, tuve dificultades para implementarla, pero considero que al implementar la cola de tareas y configurarla para distribuir las cargas de trabajo entre múltiples trabajadores, haría mucho más eficiente el proceso.
- En la cuantización aware-training también pudo haber impactado en las optimizaciones de la precisión y el tiempo de procesamiento. Sin embargo, tuve algunos errores que no me permitieron aplicar dicha cuantización.

Conclusión y futuro trabajo

LOGROS

- Las funciones para el preprocesamiento de los datos cumplían el rol que se quiso lograr en un principio.
- Si bien la red no tuvo una buena precisión, se logró crear una arquitectura de red neuronal para los dos problemas en la detección de objetos: clasificación para identificar el objeto que se encuentra en la imagen, así como el problema de regresión para que la red pueda predecir los valores de los bounding boxes para el objeto presente en la imagen.
- Se pudo asignar una tarea a cada proceso, funcionando así el paralelismo a nivel de datos.
- El tiempo de entrenamiento usando GPU fue mejorado.
- El espacio que ocupa en memoria el modelo se logró, al aplicar la cuantización post-training al modelo.

MEJORAS

- Lo que podría mejorar es implementar la cola de tareas. Tuve dificultades para implementarlo, pero considero que al implementar la cola de tareas y configurarla para distribuir las cargas de trabajo entre múltiples trabajadores, haría mucho más eficiente el proceso.
- La cuantización “aware-training” también pudo haber impactado en las optimizaciones de la precisión y el tiempo de procesamiento. Sin embargo, tuve algunos errores que no me permitieron aplicar dicha cuantización.