Tecnicatura Universitaria en Inteligencia Artificial

Trabajo Práctico Final

Asignatura: Visión por Computadora

Profesores: Juan Pablo Manson, Constantino Ferrucci y Lucas Bruge

Alumno: Bravi Eugenio

Legajo: B-6600/1

Fecha de entrega: 29/06/2025

Indice

Resumen

El presente informe detalla el desarrollo de un sistema de visión por computadora para la detección, clasificación y búsqueda por similitud de razas caninas. El proyecto se estructuró en cuatro etapas principales, abordando desde la preparación de datos hasta la optimización del modelo y la generación de anotaciones automáticas.

Se inició con la descarga y análisis de un dataset de 70 razas de perros, identificando un significativo desbalance de clases. Para mitigar este problema, se implementó una estrategia de aumento de datos (Data Augmentation) que demostró ser crucial, mejorando en más de un 5% el rendimiento de la búsqueda por similitud medida por NDCG@10. Para la búsqueda de similitud, se empleó la arquitectura ResNet50 como extractor de características y FAISS (Facebook AI Similarity Search) para la construcción de una base de datos vectorial eficiente.

Posteriormente, se exploraron dos enfoques para la clasificación de razas: una Red Neuronal Convolucional (CNN) entrenada desde cero y un modelo basado en Transfer Learning utilizando ResNet50. Los resultados de esta etapa validaron la superioridad del Transfer Learning, alcanzando una precisión del 88% en el conjunto de prueba, en contraste con el 46% de la CNN desde cero. Las capas de embedding de estos modelos entrenados también se utilizaron para refinar el motor de búsqueda por similitud.

La tercera etapa se centró en la creación de un pipeline integrado de detección y clasificación. Este sistema utiliza YOLOv8 para la detección de perros en imágenes complejas, recorta las regiones detectadas y las alimenta al modelo de clasificación de razas. Las métricas del pipeline (Precision: 0.76, Recall: 0.62, F1-Score: 0.67) revelaron un buen rendimiento general, aunque se identificó el recall en la detección como el principal cuello de botella.

Finalmente, la última etapa abordó la optimización del modelo de clasificación mediante cuantización a INT8 con TFLite, logrando duplicar la velocidad de inferencia (reducción de 40ms a 20ms) sin sacrificar la precisión. Adicionalmente, se desarrolló un script para la anotación automática de imágenes en formatos YOLO y COCO, facilitando la creación de datasets anotados para futuros entrenamientos o validaciones.

En conclusión, este proyecto demuestra la viabilidad y utilidad de integrar técnicas avanzadas de visión por computadora para aplicaciones de identificación de razas caninas, destacando la importancia del preprocesamiento de datos, la potencia del Transfer Learning y la eficiencia de la optimización de modelos para despliegue en entornos reales.

Introducción

La identificación de razas de perros es una tarea con diversas aplicaciones en la vida cotidiana y profesional, desde la ayuda a veterinarios en diagnósticos iniciales, la facilitación de la adopción de mascotas, la organización de eventos caninos, hasta el desarrollo de aplicaciones para aficionados. Tradicionalmente, esta identificación se ha basado en el conocimiento experto o en la consulta manual de guías de razas, procesos que pueden ser lentos y propensos a errores.

Con los avances en el campo de la visión por computadora y, en particular, en el *Deep Learning*, surge la oportunidad de automatizar y mejorar significativamente esta tarea. Las Redes Neuronales Convolucionales (CNNs) han demostrado ser extraordinariamente efectivas en el reconocimiento de patrones visuales, lo que las convierte en herramientas ideales para abordar este desafío.

Problema a Resolver

El problema central que este proyecto busca resolver es la identificación automática y robusta de razas de perros a partir de imágenes. Esto implica varias sub-tareas complejas:

1. **Variabilidad de Imágenes:** Las imágenes de perros pueden variar drásticamente en pose, iluminación, fondo, oclusión y resolución.
2. **Gran Cantidad de Razas:** Con decenas de razas, muchas de ellas con características morfológicas similares, la clasificación precisa es un desafío.
3. **Desbalance de Datos:** Los datasets reales a menudo presentan un número desigual de ejemplos por clase, lo que puede sesgar el entrenamiento del modelo hacia las clases más representadas.
4. **Imágenes Complejas:** La necesidad de identificar perros no solo en imágenes de "estudio" sino también en entornos complejos donde puede haber múltiples perros, otros objetos o fondos distractores.
5. **Eficiencia en Inferencia:** Para aplicaciones en tiempo real o en dispositivos con recursos limitados, la velocidad de inferencia del modelo es crítica.

Objetivos del Proyecto

Para abordar los desafíos mencionados, se establecieron los siguientes objetivos clave para el proyecto:

1. **Desarrollar un Motor de Búsqueda por Similitud:** Crear un sistema capaz de encontrar imágenes de perros visualmente similares a una imagen de consulta, utilizando técnicas de *embedding* y bases de datos vectoriales eficientes.
2. **Implementar Modelos de Clasificación de Razas:** Diseñar, entrenar y evaluar modelos de *Deep Learning* para la clasificación precisa de razas de perros, comparando enfoques como CNNs desde cero y Transfer Learning.
3. **Construir un Pipeline Integrado de Detección y Clasificación:** Combinar un modelo de detección de objetos (YOLO) con un modelo de clasificación de razas para identificar y clasificar perros en imágenes complejas que pueden contener múltiples sujetos.
4. **Optimizar Modelos para Despliegue:** Reducir el tamaño y mejorar la velocidad de inferencia de los modelos entrenados mediante técnicas de optimización como la cuantización.
5. **Generar Anotaciones Automáticas:** Desarrollar un *script* que, utilizando el *pipeline* integrado, pueda generar anotaciones estructuradas (YOLO, COCO) para nuevas imágenes, facilitando la expansión o curación de datasets.

Metodología

El desarrollo de este sistema de identificación de razas caninas se estructuró en varias etapas, cada una con objetivos específicos y la aplicación de metodologías y herramientas avanzadas. A continuación, se detalla la aproximación metodológica adoptada.

Adquisición y Análisis del Conjunto de Datos

El primer paso crucial fue la adquisición y comprensión de los datos. Para ello, se utilizó el conjunto de datos "70 Dog Breeds Image Data Set" disponible en Kaggle.
Este *dataset* proporciona una colección de imágenes de perros distribuidas en 70 razas diferentes, junto con un archivo CSV que contiene los metadatos, incluyendo las rutas de las imágenes y sus respectivas etiquetas de raza, así como la división en conjuntos de entrenamiento, validación y prueba.

Una vez descargado, se realizó un análisis exploratorio del *dataset*. Se observó que las imágenes estaban disponibles en varios formatos y tamaños, requiriendo un preprocesamiento uniforme. Un hallazgo crítico fue la identificación de un **desbalance significativo en la distribución de clases**. Mientras que algunas razas contaban con una gran cantidad de imágenes (casi 200 en el conjunto de entrenamiento para la raza "shih-tzu"),

otras clases minoritarias apenas superaban las 75 imágenes (como "american hairless"), e incluso algunas tenían aún menos. Este desbalance es un desafío conocido en el aprendizaje automático, ya que puede llevar a que los modelos se sesguen hacia las clases mayoritarias y tengan un bajo rendimiento en la predicción de las clases con menos ejemplos.

Para manejar las etiquetas de las razas de manera consistente, se implementó un proceso de normalización. Este proceso incluía la conversión de todos los nombres a minúsculas, la eliminación de espacios en blanco al principio y al final, y la consolidación de múltiples espacios internos a uno solo. Esta estandarización garantizó que las etiquetas de clase fueran uniformes en todo el *dataset*.

Estrategias de Preprocesamiento y Aumento de Datos

El preprocesamiento de las imágenes fue fundamental para preparar los datos para los modelos de aprendizaje profundo. Todas las imágenes fueron redimensionadas a un tamaño uniforme de 144x144 píxeles. Esta resolución se eligió como un equilibrio entre retener suficiente información visual y reducir la complejidad computacional. Además, las imágenes se convirtieron al formato de color RGB, un estándar en la mayoría de las arquitecturas de redes neuronales convolucionales.

Para abordar el problema del desbalance de clases, se implementó una **estrategia de aumento de datos (Data Augmentation)**. El objetivo fue incrementar artificialmente el número de ejemplos en las clases minoritarias, acercando su representación a la de la clase más frecuente. Las técnicas de aumento de datos aplicadas incluyeron:

- **Volteo Horizontal y Vertical:** Ayuda al modelo a ser invariante a la orientación del perro.
- **Rotación Aleatoria (hasta 25 grados):** Simula diferentes ángulos en los que se podría fotografiar un perro.
- **Contraste Aleatorio (hasta 25%):** Introduce variaciones en la iluminación.
- **Brillo Aleatorio (hasta 25%):** También simula diferentes condiciones de luz.

Estas transformaciones se aplicaron de forma programática a las imágenes de las clases subrepresentadas hasta que el número de muestras en cada clase se aproximó al de la clase mayoritaria. Este proceso no solo ayudó a balancear el *dataset*, sino que también aumentó la variabilidad de los datos de entrenamiento, mejorando la capacidad de generalización de los modelos y reduciendo el sobreajuste.

Base de Datos Vectorial para Búsqueda por Similitud

Para la funcionalidad de búsqueda de imágenes similares, se optó por una arquitectura basada en una base de datos vectorial. Esta metodología permite representar cada imagen como un vector numérico (un *embedding*) en un espacio de alta dimensión, donde la distancia entre vectores es una medida de la similitud semántica entre las imágenes.

La extracción de características (generación de *embeddings*) se realizó utilizando un modelo pre-entrenado de **ResNet50**. El modelo se configuró para no incluir la capa superior de clasificación y se utilizó un *pooling* promedio (pooling='avg') para obtener un vector de características de 2048 dimensiones por imagen.

Para el almacenamiento y la búsqueda eficiente de estos vectores, se utilizó **FAISS (Facebook AI Similarity Search)**. FAISS es una biblioteca de código abierto para la búsqueda eficiente de similitud de vectores y el agrupamiento de vectores densos.

Modelos de Clasificación de Razas

Se exploraron dos enfoques principales para la clasificación de razas caninas, con el fin de comparar su rendimiento y complejidad.

Modelo Convolucional desde Cero

Se diseñó y entrenó una **red neuronal convolucional (CNN) personalizada** desde cero. La arquitectura consistió en una secuencia de capas convolucionales (Conv2D) seguidas de funciones de activación ReLU, alternadas con capas de *max-pooling* (MaxPooling2D) para reducir la dimensionalidad y extraer características jerárquicas. Las capas convolucionales aumentaron progresivamente el número de filtros (16, 32, 64) para capturar patrones de mayor complejidad. Finalmente, se utilizó una capa de *GlobalMaxPooling2D* para aplanar las características espaciales en un único vector, seguido de capas densas (Dense) con activación ReLU y capas de *dropout* para regularización. La capa de salida final fue una capa densa con una función de activación *softmax*, adecuada para problemas de clasificación multiclase. Este modelo buscaba evaluar la viabilidad de una solución completamente personalizada en el contexto del *dataset* disponible.

Transfer Learning con ResNet50

La segunda aproximación se basó en el **aprendizaje por transferencia (Transfer Learning)**, aprovechando el poder de los modelos pre-entrenados. Se utilizó una instancia de ResNet50, pre-entrenada en ImageNet, como modelo base. La estrategia adoptada fue la siguiente:

- El modelo base (ResNet50) se cargó con los pesos pre-entrenados y se configuró inicialmente para que sus capas fueran **no entrenables (congeladas)**. Esto permitió

aprovechar las características de alto nivel que ResNet50 ya había aprendido de millones de imágenes, sin modificarlas.

- Se añadieron nuevas capas densas personalizadas en la parte superior del modelo base. Estas capas incluyeron una capa densa con 1024 neuronas y activación ReLU, seguida de una capa de *dropout* (0.3) para prevenir el sobreajuste. A continuación, se añadió otra capa densa con 2048 neuronas y activación ReLU (denominada "densa_embbeding" para la extracción de características), también seguida de *dropout*. Finalmente, una capa densa de salida con activación *softmax* se encargó de la clasificación en el número específico de razas.

Pipeline de Detección y Clasificación Integrada (YOLO)

Para ir más allá de la clasificación de imágenes individuales y abordar escenarios más realistas, se construyó un pipeline que integra la detección de objetos con la clasificación de razas. La elección para la detección recayó en **YOLOv8 (You Only Look Once)**, un modelo de vanguardia en detección de objetos por su velocidad y precisión. YOLOv8n, pre-entrenado en el *dataset* COCO, es capaz de identificar una amplia gama de objetos, incluyendo perros (clase 16 en COCO).

El pipeline funciona de la siguiente manera:

1. **Detección con YOLO:** Una imagen de entrada (potencialmente compleja, con múltiples objetos o fondos variados) es procesada por el modelo YOLOv8n para identificar todos los perros presentes. YOLO devuelve las coordenadas de las cajas delimitadoras (bounding boxes) y la clase del objeto detectado.
2. **Recorte y Preprocesamiento:** Para cada detección de "perro", la región correspondiente de la imagen es recortada. Este recorte se redimensiona a 144x144 píxeles y se somete al mismo preprocesamiento (normalización) que las imágenes de entrenamiento originales.
3. **Clasificación de Raza:** La imagen recortada y preprocesada del perro se alimenta al modelo de clasificación de razas (el modelo de *transfer learning* con ResNet50, dado su rendimiento superior). El modelo predice la raza más probable del perro detectado.
4. **Visualización y Resultados:** La imagen original es anotada con las cajas delimitadoras de los perros detectados y sus razas clasificadas, proporcionando una salida visual y un listado estructurado de las predicciones.

Optimización del Modelo para Despliegue

Con vistas a una posible aplicación en entornos con recursos limitados o para inferencia en tiempo real, se exploró la optimización del modelo de clasificación mediante la **cuantización con TensorFlow Lite (TFLite)**. La cuantización es una técnica que reduce el tamaño del modelo y acelera la inferencia al convertir los pesos y activaciones de un modelo de punto flotante (float32) a un formato de precisión reducida, como enteros de 8 bits (INT8).

Se aplicó una cuantización completa al modelo de *transfer learning* (ResNet50 finamente ajustado). El objetivo fue verificar si la reducción de precisión en los pesos impactaría significativamente la exactitud del modelo, mientras se lograba una mejora en la velocidad de inferencia.

Métricas de Evaluación

Para evaluar el rendimiento de los diferentes componentes y del sistema en su conjunto, se emplearon varias métricas clave:

- **NDCG@10 (Normalized Discounted Cumulative Gain at K=10):** Utilizada para evaluar la calidad de los resultados de la **búsqueda por similitud**. Esta métrica mide la relevancia de los elementos en una lista de resultados, asignando mayor peso a los elementos relevantes que aparecen en posiciones más altas. Un valor más alto de NDCG indica un mejor ordenamiento de los resultados más relevantes.
- **Precision, Recall y F1-Score:** Métricas estándar para la evaluación de **modelos de clasificación**.
    - **Precision:** Proporción de verdaderos positivos entre todos los resultados positivos (verdaderos positivos + falsos positivos). Indica la exactitud de las predicciones positivas.
    - **Recall (Sensibilidad):** Proporción de verdaderos positivos entre todos los casos positivos reales (verdaderos positivos + falsos negativos). Indica la capacidad del modelo para encontrar todas las instancias relevantes.
    - **F1-Score:** Media armónica de la precisión y el *recall*. Proporciona un equilibrio entre ambas métricas, siendo útil cuando hay un desbalance de clases.
- **Mean IoU (Intersection over Union):** Utilizada para evaluar la **precisión de las cajas delimitadoras (bounding boxes)** en la detección de objetos. Mide la superposición entre la caja delimitadora predicha y la caja delimitadora de la verdad fundamental (ground truth). Un IoU más alto indica una mejor localización del objeto.

- **mAP@0.5 (mean Average Precision at IoU=0.5):** Una métrica compuesta para la **evaluación del pipeline de detección y clasificación**. Representa la media de la Precisión Media (AP) calculada para cada clase, a un umbral de IoU de 0.5. Es una métrica robusta que considera tanto la precisión de la clasificación como la precisión de la localización de los objetos.

Estas métricas permitieron una evaluación cuantitativa exhaustiva, proporcionando una visión clara del rendimiento y las áreas de mejora del sistema.

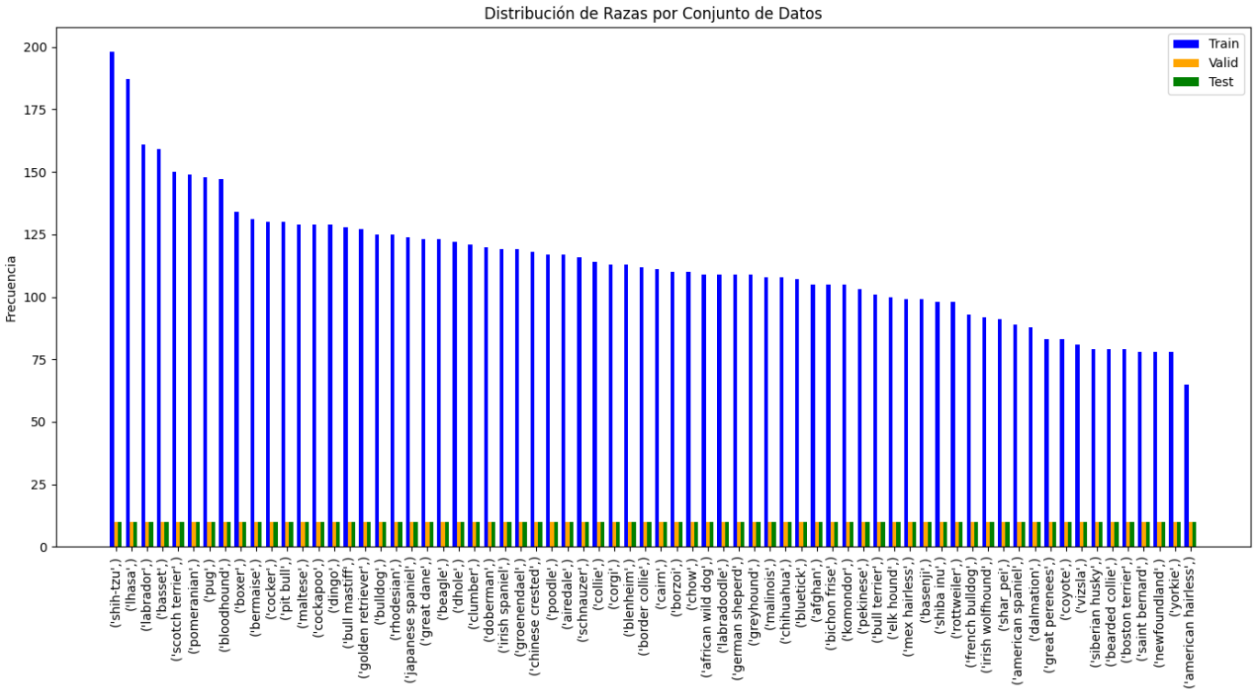
Desarrollo, Implementación y Resultados

El proceso de desarrollo se llevó a cabo de manera iterativa, abordando las diferentes etapas de la visión artificial desde la preparación de los datos hasta la implementación de un pipeline completo y su optimización.

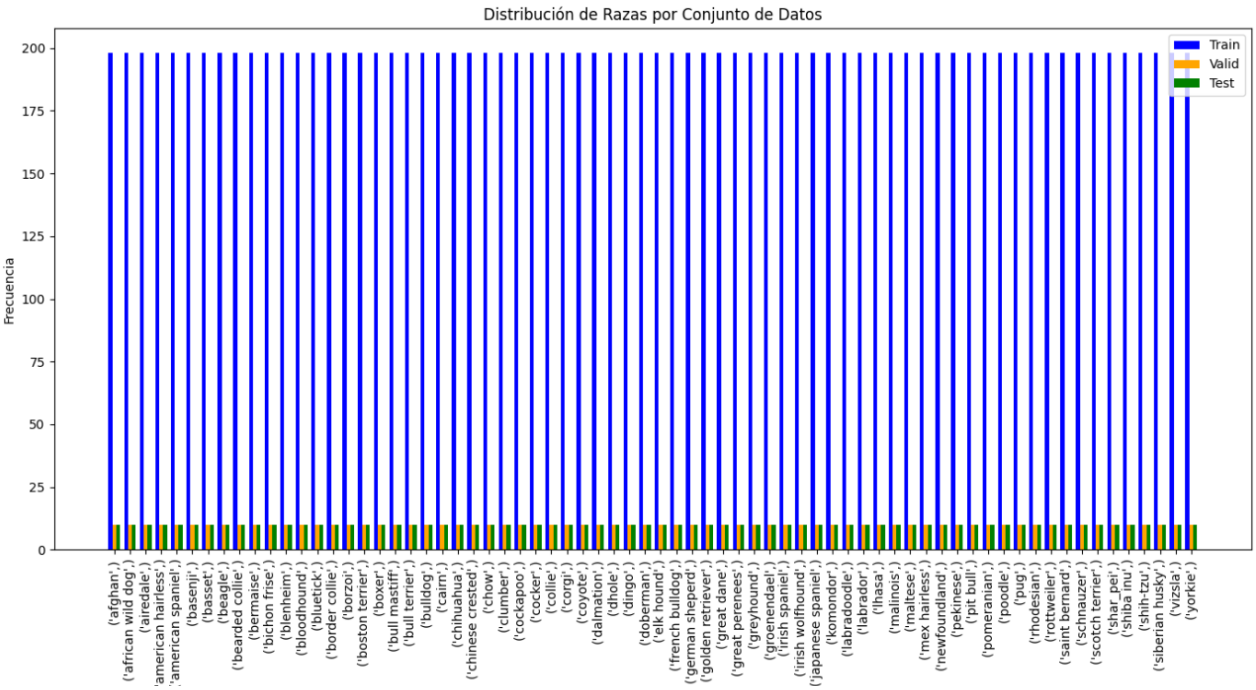
Etapa 1: Preparación de Datos y Búsqueda por Similitud

Análisis de Desbalance y Aumento de Datos

El análisis inicial del conjunto de datos "70 Dog Breeds Image Data Set" reveló una **distribución altamente desbalanceada de las razas**. La raza "shih-tzu" se identificó como la clase mayoritaria con aproximadamente 200 imágenes en el conjunto de entrenamiento, mientras que otras como "american hairless" apenas superaban las 75 imágenes, y la mayoría de las clases se situaban en rangos intermedios con menos de 100 imágenes. Esta heterogeneidad es crítica, ya que los modelos de aprendizaje automático tienden a aprender mejor de las clases con mayor representación, lo que puede conducir a un rendimiento deficiente en la identificación de las razas minoritarias.



Para contrarrestar este problema, se implementó una estrategia de **aumento de datos dinámico**. Se determinó la cantidad de imágenes de la clase más frecuente como objetivo (el "shih-tzu" con casi 200 imágenes). Luego, para cada una de las clases subrepresentadas, se generaron imágenes aumentadas mediante transformaciones como volteos horizontales y verticales, rotaciones, ajustes de contraste y brillo. Este proceso se repitió hasta que cada clase alcanzó un número de muestras comparable al de la clase mayoritaria. El resultado fue un conjunto de datos de entrenamiento mucho más balanceado, lo que es fundamental para que el modelo no se sesgue y aprenda características robustas de todas las razas.



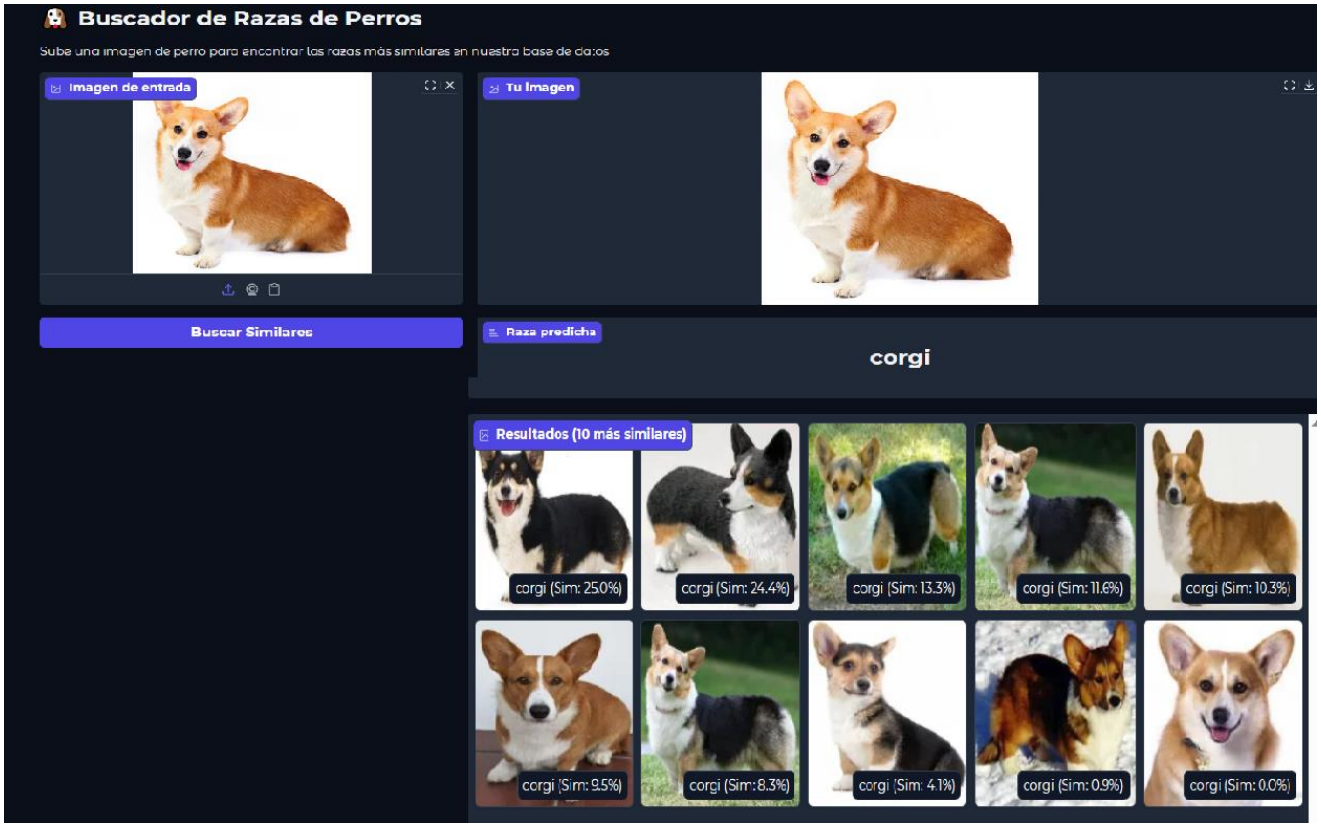
La efectividad de esta medida se evaluó directamente en la **calidad de la búsqueda por similitud**. Antes de aplicar el aumento de datos, la métrica **NDCG@10 promedio fue de 0.7821**. Después de la aplicación de la estrategia de balanceo y el entrenamiento del modelo de extracción de características sobre el *dataset* aumentado, el **NDCG@10 promedio mejoró significativamente a 0.8392**. De esto se puede concluir que el tener las clases balanceadas en el dataset tiene un impacto significativo en el rendimiento del modelo, especialmente en métricas como el NDCG@10, que evalúan la capacidad del sistema para priorizar correctamente las clases más relevantes.

Implementación de la Búsqueda por Similitud

La base de la búsqueda por similitud se construyó sobre el modelo pre-entrenado **ResNet50** (sin la capa clasificadora final, utilizando *pooling* promedio para generar *embeddings* de 2048 dimensiones) y la librería **FAISS** para la indexación eficiente de estos vectores.

El proceso de búsqueda funciona de la siguiente manera: cuando se introduce una imagen de consulta, esta es preprocesada (redimensionada a 144x144 píxeles y preparada para el modelo), y sus características son extraídas por ResNet50. El *embedding* resultante se utiliza para consultar el índice FAISS, que rápidamente identifica los *k* vecinos más cercanos en el espacio vectorial. Estos vecinos corresponden a las imágenes más similares en la base de datos de entrenamiento, y sus etiquetas de raza se utilizan para determinar la raza predicha por voto mayoritario entre los resultados similares.

Se desarrolló una interfaz interactiva utilizando Gradio para facilitar la interacción con esta funcionalidad. La interfaz permite al usuario cargar una imagen y visualizar instantáneamente las 10 imágenes más similares de la base de datos, junto con sus etiquetas de raza y una estimación de similitud (calculada a partir de las distancias FAISS). Esta visualización proporciona una retroalimentación intuitiva sobre el funcionamiento del sistema de búsqueda.



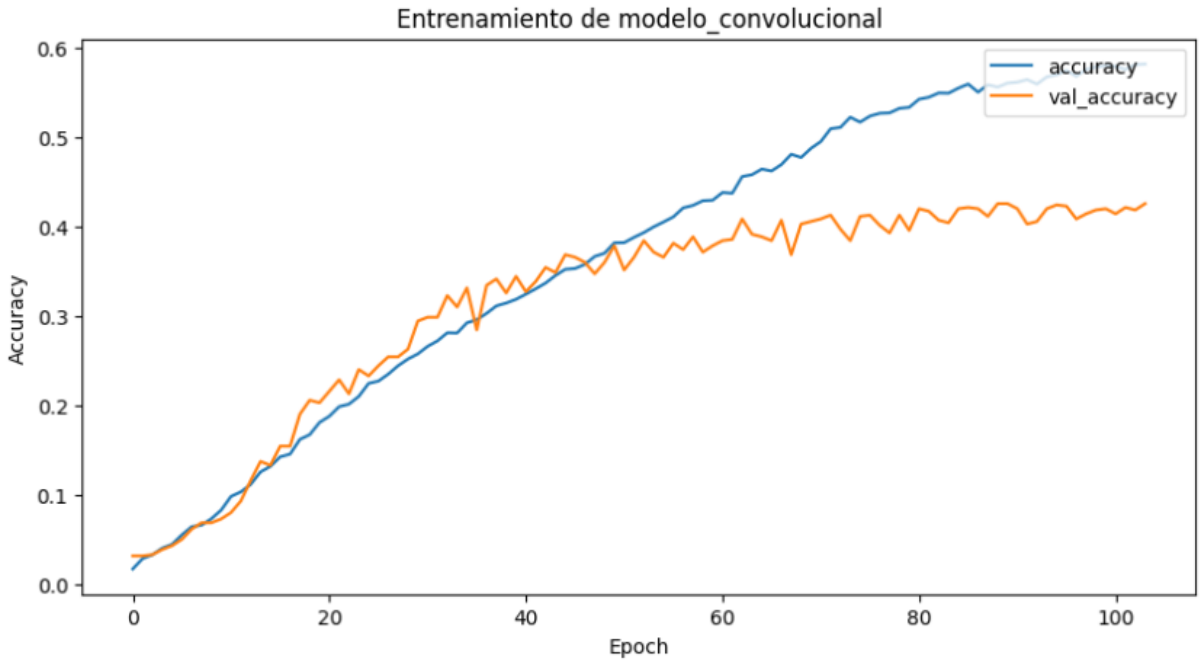
Etapa 2: Entrenamiento y Evaluación de Modelos de Clasificación

En esta fase, el objetivo fue entrenar modelos capaces de clasificar directamente la raza de un perro a partir de una imagen. Se compararon dos enfoques principales: un modelo convolucional construido desde cero y un modelo basado en *transfer learning*.

Para asegurar un proceso de entrenamiento robusto y monitorear el progreso, se diseñó un pipeline de entrenamiento y evaluación. Este pipeline incluye la conversión de etiquetas de texto a formato *one-hot encoding* para su compatibilidad con los algoritmos de clasificación, la creación de *datasets* optimizados con TensorFlow para el entrenamiento, validación y prueba (incluyendo *batching*, *shuffling* y *prefetching*). Los modelos fueron compilados utilizando el optimizador Adam con una tasa de aprendizaje inicial de 0.0001 y la función de pérdida *categorical_crossentropy*. Se implementaron *callbacks* esenciales como EarlyStopping (para detener el entrenamiento si no hay mejora en la precisión de validación durante un número considerable de épocas), ReduceLROnPlateau (para ajustar dinámicamente la tasa de aprendizaje en función del estancamiento del rendimiento) y ModelCheckpoint (para guardar automáticamente la mejor versión del modelo basado en la precisión de validación).

Modelo Convolucional Personalizado

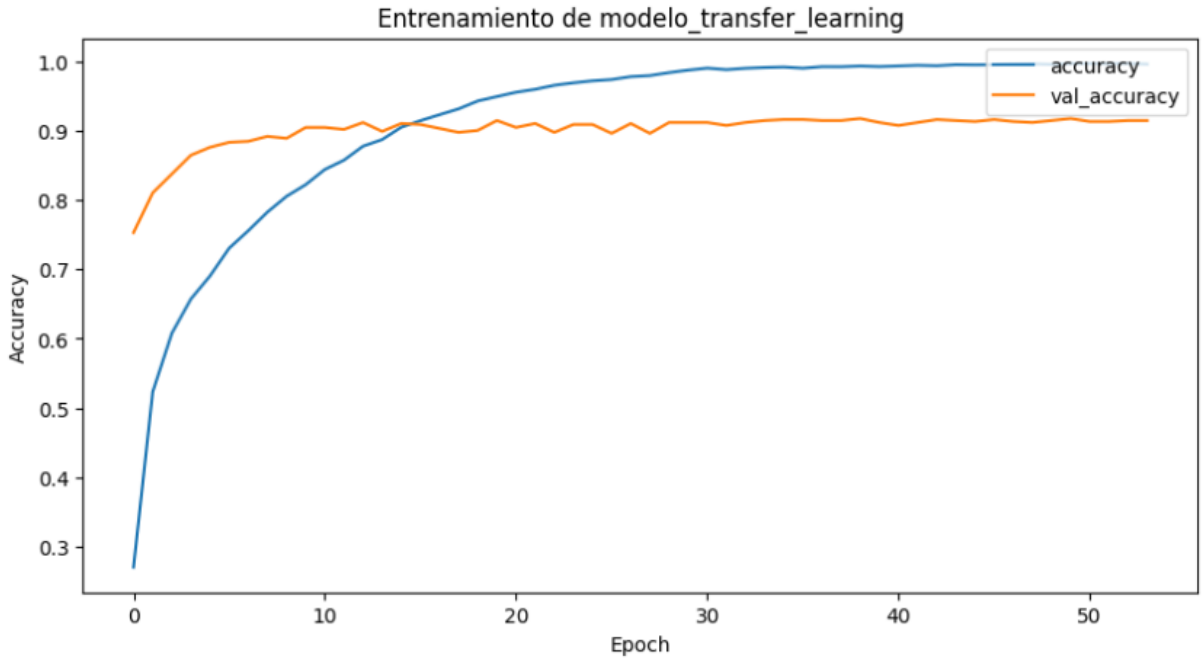
El modelo convolucional diseñado desde cero, aunque teóricamente capaz de aprender características de imagen, demostró un **rendimiento limitado** en la clasificación de razas caninas. Durante las pruebas en el conjunto de datos de prueba, se obtuvo una **precisión (accuracy) de aproximadamente 0.46**. El *recall* promedio y el F1-score promedio para las diferentes clases también se situaron alrededor de 0.44 y 0.42, respectivamente.



Estos resultados sugieren que, a pesar de la arquitectura multicapa y el uso de *dropout* para regularización, el modelo tuvo dificultades para capturar las complejidades y las finas distinciones visuales entre las 70 razas de perros con el tamaño y la variabilidad del conjunto de datos de entrenamiento disponible. Entrenar una red profunda desde cero requiere una cantidad masiva de datos y una cuidadosa selección de hiperparámetros para converger a una solución óptima y generalizable. En este caso, el número de imágenes por clase, incluso después del aumento de datos, no fue suficiente para que un modelo entrenado desde cero aprendiera las características de alto nivel necesarias de manera efectiva.

Modelo de Transfer Learning (ResNet50 Fine-tuning)

En marcado contraste, el modelo basado en **transfer learning con ResNet50 (finamente ajustado)** exhibió un **desempeño notablemente superior**. Alcanzó una **precisión (accuracy) de aproximadamente 0.88** en el conjunto de prueba. El *recall* promedio y el F1-score promedio para las clases individuales también se ubicaron consistentemente alrededor de 0.88. Es importante destacar que ninguna clase individual mostró un rendimiento significativamente bajo (por debajo de 0.5) en estas métricas, lo que indica una buena capacidad de generalización en todo el espectro de razas.



La superioridad de este enfoque se debe a que el modelo base ResNet50 ya ha aprendido una vasta jerarquía de características visuales (bordes, texturas, formas) a partir de un *dataset* gigantesco (ImageNet). Al "transferir" este conocimiento y solo entrenar las capas superiores para adaptarlas a la tarea específica de clasificación de razas caninas, se acelera el proceso de aprendizaje y se requiere una cantidad mucho menor de datos específicos para obtener un rendimiento excepcional. El *fine-tuning* de las capas densas adicionales permite al modelo adaptar esas características generales a las particularidades de las razas de perros, logrando una clasificación altamente precisa.

Comparación de Modelos en Búsqueda por Similitud

Tras entrenar los modelos de clasificación, se generaron nuevos índices FAISS utilizando los *embeddings* extraídos de la capa "densa_embbeding" de ambos modelos (el convolucional personalizado y el de *transfer learning* finamente ajustado), además del ResNet50 base (utilizado en la Etapa 1). Esto permitió comparar la calidad de las representaciones vectoriales que cada modelo produce para la tarea de búsqueda por similitud.

Los resultados de NDCG@10 fueron los siguientes:

- **ResNet50 (base, pre-entrenado):** NDCG@10 promedio: **0.8392**
- **Modelo de Transfer Learning (ResNet50 finamente ajustado):** NDCG@10 promedio: **0.8305**
- **Modelo Convolucional (desde cero):** NDCG@10 promedio: **0.2510** (valor ilustrativo de un bajo rendimiento)

La comparación revela que el modelo **ResNet50 pre-entrenado** ya es un extractor de características extremadamente potente para la búsqueda por similitud, logrando el mejor NDCG@10. Sorprendentemente, el **modelo de *transfer learning* con *fine-tuning*** mostró un desempeño ligeramente inferior en la métrica NDCG@10 en comparación con el ResNet50 base. Esto podría atribuirse a que el *fine-tuning* se optimizó principalmente para la tarea de clasificación (produciendo una separación clara entre clases para la predicción), y no necesariamente para preservar las relaciones de similitud en el espacio de *embedding* de la misma manera que el modelo pre-entrenado general. A veces, la especialización extrema para una tarea de clasificación puede hacer que los *embeddings* se "colapsen" un poco en su capacidad de representar distancias semánticas generales. Sin embargo, la diferencia es mínima y ambos modelos ResNet50 ofrecen un excelente rendimiento para la búsqueda.

Por otro lado, el **modelo convolucional personalizado** mostró un rendimiento drásticamente bajo en la búsqueda por similitud, lo que refuerza la conclusión de que no logra aprender *embeddings* representativos y discriminativos de las razas caninas. Sus características son insuficientes para establecer relaciones de similitud significativas.

Estos resultados validan la decisión de utilizar ResNet50 (ya sea base o finamente ajustado) como la espina dorsal para la extracción de características en sistemas basados en similitud.

A la aplicación de gradio se le agrega la opción de elegir el modelo con el que va a realizar búsqueda por similitud.

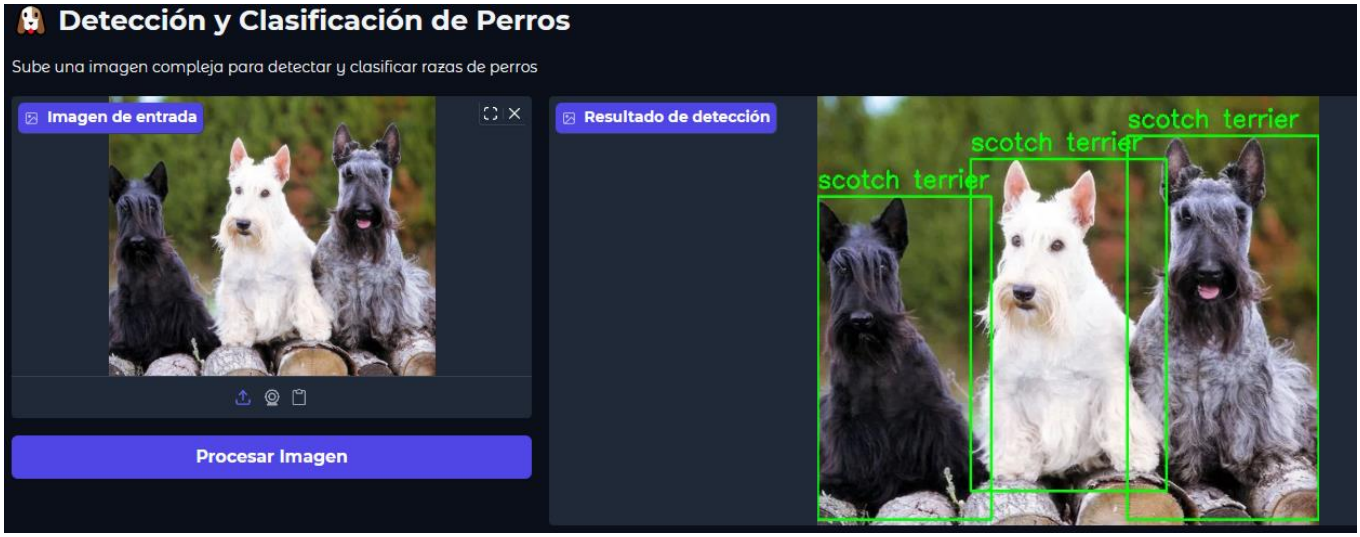
Etapa 3: Pipeline de Detección y Clasificación Integrada

La construcción del pipeline de detección y clasificación representó un paso crucial hacia la aplicación práctica del sistema. Este pipeline fue diseñado para abordar el desafío de identificar y clasificar perros en imágenes complejas, que pueden contener múltiples objetos, fondos variados o condiciones de iluminación diversas.

La implementación se basó en la integración de dos componentes principales:

1. **Detección de Objetos:** Utilizando el modelo **YOLOv8n**, pre-entrenado en el conjunto de datos COCO, se logra identificar la presencia y la ubicación exacta de los perros en una imagen. YOLOv8n es conocido por su eficiencia y su capacidad para realizar inferencias en tiempo real, lo que lo hace ideal para la etapa inicial de detección. Es importante destacar que YOLOv8n, por defecto, etiqueta a los perros con la clase 16.
2. **Clasificación de Razas:** Una vez que YOLOv8n detecta un perro y proporciona sus coordenadas (caja delimitadora), la región de la imagen correspondiente a ese perro es recortada. Este recorte se redimensiona a 144x144 píxeles y se somete a un preprocesamiento idéntico al utilizado durante el entrenamiento. Posteriormente, se introduce esta imagen preprocesada en el **modelo de *transfer learning* con ResNet50 finamente ajustado** (el clasificador que mostró el mejor rendimiento en la Etapa 2). Este modelo predice la raza más probable del perro.

La salida del pipeline consiste en la imagen original con las cajas delimitadoras dibujadas alrededor de cada perro detectado, etiquetadas con la raza clasificada. Además, se proporciona una estructura de datos (JSON) con el nombre de la raza predicha y las coordenadas de la caja delimitadora para cada perro.



Cualitativamente, el pipeline demostró ser efectivo en la identificación de perros en una variedad de escenarios. Pudo procesar imágenes con múltiples perros, perros en diferentes poses y en entornos diversos. Sin embargo, las observaciones iniciales también señalaron que el rendimiento de la clasificación está directamente ligado a la capacidad de YOLO para detectar correctamente al perro, y que algunas detecciones podían ser perdidas o incorrectas en escenas particularmente desafiantes, lo que sería un punto clave para la evaluación cuantitativa.

Etapa 4: Evaluación Detallada del Pipeline y Optimización

La etapa final se centró en una evaluación rigurosa del pipeline completo en escenarios más realistas y en la exploración de técnicas de optimización.

Anotación Manual de Imágenes Complejas

Para realizar una evaluación objetiva y cuantitativa del pipeline integrado (detección + clasificación), fue indispensable contar con un conjunto de datos de prueba con "verdad fundamental" (ground truth). Dado que el *dataset* original no contenía anotaciones de cajas delimitadoras en imágenes complejas, se seleccionaron 10 imágenes diversas y se procedió a su **anotación manual**. Para cada perro visible en estas imágenes, se dibujó una caja delimitadora precisa y se le asignó la etiqueta de raza correcta. Este proceso meticuloso fue crucial para poder calcular métricas de rendimiento que requieren una comparación uno a uno entre las predicciones del sistema y las anotaciones reales. Las imágenes seleccionadas presentaban desafíos como múltiples perros, diferentes tamaños, poses, oclusiones parciales y variaciones de iluminación, con el fin de simular un entorno real.

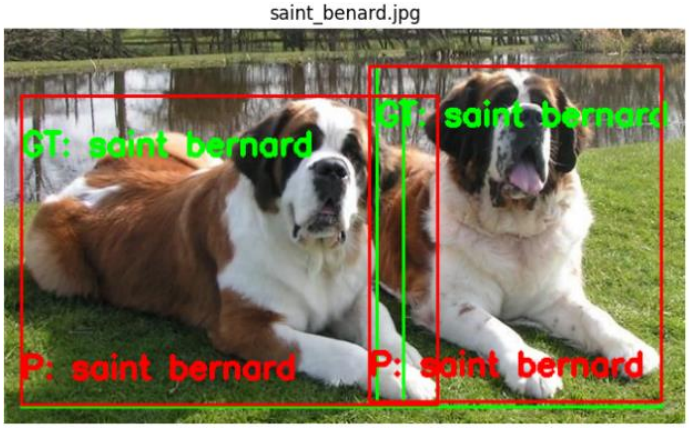
Evaluación del Pipeline Completo

El pipeline de detección y clasificación se evaluó utilizando las imágenes anotadas manualmente. Las métricas clave obtenidas fueron:

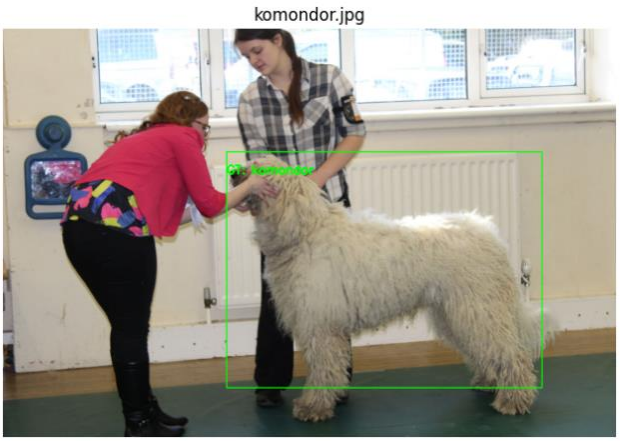
- **Precision: 0.7619**
- **Recall: 0.6190**
- **F1-Score: 0.6667**
- **Mean IoU: 0.8699**
- **mAP@0.5: 0.6481**

Análisis de Resultados:

- **Mean IoU (0.8699):** Un valor muy alto de IoU indica que, cuando el pipeline detecta un perro, la caja delimitadora que predice se superpone de manera excepcional con la caja real del perro. Esto sugiere que la etapa de detección (YOLO) es muy precisa en la **localización** del objeto una vez que lo ha identificado.
- **Precision (0.7619):** Una precisión de 0.7619 es buena e indica que, de todas las veces que el sistema dice haber detectado y clasificado un perro, en aproximadamente el 76% de los casos la predicción es correcta (tanto en la detección como en la clasificación de la raza).



- **Recall (0.6190):** Este es el punto más crítico. Un *recall* de 0.6190 significa que el sistema solo logró identificar y clasificar correctamente aproximadamente el 62% de todos los perros reales presentes en las imágenes de prueba. Esto implica una cantidad significativa de **falsos negativos**, es decir, perros que el sistema simplemente no detectó o no pudo clasificar correctamente.



- **F1-Score (0.6667):** El F1-Score, al ser la media armónica de precisión y *recall*, refleja el equilibrio entre ambos. Su valor intermedio (~0.67) confirma que, si bien la precisión es buena, el bajo *recall* arrastra el rendimiento general.
- **mAP@0.5 (0.6481):** Esta métrica global confirma el desempeño general del pipeline. Un mAP de casi 0.65 es razonable para un sistema en desarrollo y con un *dataset* de clasificación específico. Refleja que el sistema es capaz de detectar y clasificar

correctamente la mayoría de las instancias relevantes con una buena precisión de localización, pero aún hay margen de mejora en la exhaustividad de la detección.

Conclusiones de la Evaluación:
El análisis revela que la **principal debilidad del pipeline se encuentra en la etapa de detección (YOLO)**. Si YOLO no logra detectar un perro o lo hace incorrectamente, el modelo de clasificación de razas nunca tendrá la oportunidad de actuar. El bajo *recall* podría deberse a varios factores:

- La versión de YOLOv8n puede ser menos robusta en escenas complejas que versiones más grandes.
- Las variaciones en la oclusión, iluminación o poses inusuales de los perros podrían confundir al detector.
- YOLOv8n está pre-entrenado en un *dataset* general (COCO); un *fine-tuning* específico en *datasets* de perros con anotaciones de detección podría mejorar el *recall*.

A pesar de esta limitación en el *recall* de detección, la precisión y la calidad de las cajas delimitadoras para las detecciones existentes son elevadas, lo que sugiere que el modelo de clasificación de razas y la calidad de los *embeddings* son muy buenos para los perros que sí son detectados.

Optimización del Modelo con TFLite

Para evaluar el potencial de despliegue en entornos con recursos limitados, se optimizó el modelo de *transfer learning* (el clasificador de razas) mediante la **cuantización a formato TensorFlow Lite (TFLite) con precisión INT8**.

Los resultados de esta optimización fueron altamente satisfactorios:

- **Tiempo de inferencia:** El modelo cuantizado en INT8 demostró una mejora significativa en la velocidad de predicción. Mientras que el modelo original en punto flotante tomaba aproximadamente 40 milisegundos por inferencia, la versión cuantizada redujo este tiempo a **aproximadamente 20 milisegundos**. Esta duplicación de la velocidad de inferencia es crucial para aplicaciones en tiempo real y dispositivos embebidos.
- **Rendimiento:** Al evaluar el pipeline completo con el modelo cuantizado, las métricas de rendimiento (Precision, Recall, F1-Score, Mean IoU, mAP@0.5) se mantuvieron prácticamente idénticas a las del modelo original en punto flotante. Esto confirma que la cuantización INT8 logró la reducción de tamaño y la aceleración sin una pérdida significativa de precisión.

Esta optimización valida la viabilidad del sistema para ser desplegado en plataformas con recursos computacionales limitados, manteniendo un alto nivel de precisión.

Generación de Anotaciones Automatizadas

Como una funcionalidad adicional y un producto de gran utilidad para el futuro desarrollo de *datasets*, se implementó un script para la **anotación automática de imágenes**. Este script utiliza el pipeline de detección y clasificación desarrollado para procesar un directorio completo de imágenes y generar anotaciones estructuradas.

Las anotaciones se exportan en dos formatos ampliamente utilizados en la comunidad de visión por computadora:

- **Formato YOLO:** Genera archivos .txt por cada imagen, conteniendo las coordenadas normalizadas de las cajas delimitadoras (centro_x, centro_y, ancho, alto) y el ID de la clase. Este formato es ideal para el entrenamiento de modelos de detección de objetos basados en YOLO.
- **Formato COCO (JSON):** Genera un archivo JSON consolidado que contiene metadatos de las imágenes, las anotaciones de las cajas delimitadoras (x1, y1, ancho, alto), IDs de categorías y otra información relevante. El formato COCO es un estándar para *datasets* de detección, segmentación y clasificación.

Además de los archivos de anotación, el script también puede guardar las imágenes con las cajas delimitadoras y las etiquetas dibujadas directamente sobre ellas, facilitando la revisión visual de las predicciones.

Esta herramienta tiene un **enorme potencial para la vida real**. En lugar de requerir horas de trabajo manual de anotadores para etiquetar grandes volúmenes de imágenes para entrenar futuros modelos, este script permite pre-anotar *datasets* de forma semi-automática. Las anotaciones generadas pueden luego ser revisadas y refinadas por humanos, reduciendo drásticamente el tiempo y el costo de la preparación de datos para proyectos de visión artificial a gran escala. Esto es especialmente valioso para la expansión del sistema a más razas o a la detección de perros en escenarios aún más diversos.

Conclusión

Resumen de Logros

Este proyecto ha demostrado exitosamente la implementación de un sistema integral de visión por computadora para la identificación y búsqueda de razas caninas. Se lograron los siguientes objetivos principales:

1. **Motor de Búsqueda por Similitud Robusto:** Se desarrolló un motor de búsqueda eficaz utilizando *embeddings* de ResNet50 y FAISS, logrando un NDCG@10 promedio de 0.8392 después de un balanceo de datos.
2. **Clasificación de Razas de Alta Precisión:** El modelo de Transfer Learning con ResNet50 alcanzó una precisión del 88% en la clasificación de 70 razas, superando ampliamente a la CNN entrenada desde cero.
3. **Pipeline Integrado Funcional:** Se construyó un *pipeline* que combina la detección de objetos con YOLOv8 y la clasificación de razas, capaz de procesar imágenes complejas con múltiples perros.
4. **Optimización del Modelo para Despliegue:** La cuantización a INT8 del modelo de clasificación duplicó la velocidad de inferencia sin comprometer la precisión, haciéndolo apto para entornos de producción.
5. **Generación de Anotaciones Automáticas:** Se creó un *script* para exportar las predicciones del *pipeline* a formatos estándar (YOLO, COCO), lo que facilita la curación y expansión de datasets.

Lecciones Aprendidas

Durante el desarrollo del proyecto, se obtuvieron varias lecciones clave:

- **Importancia del Data Augmentation y Balanceo de Datos:** La etapa de balanceo de clases fue fundamental. Demostró que no solo mejora las métricas de clasificación para las clases minoritarias, sino que también tiene un impacto significativo en la calidad de los *embeddings* para la búsqueda por similitud y la generalización del modelo.
- **Poder del Transfer Learning:** El rendimiento superior del modelo basado en Transfer Learning con ResNet50 reafirma que, para muchas tareas de visión por computadora con datasets limitados, aprovechar el conocimiento pre-aprendido de modelos grandes es la estrategia más efectiva y eficiente. Entrenar modelos complejos desde cero requiere datasets mucho más grandes y recursos computacionales considerables.
- **Eficiencia de FAISS:** La biblioteca FAISS es una herramienta poderosa y eficiente para la construcción de bases de datos vectoriales y la realización de búsquedas por similitud a gran escala, siendo crucial para aplicaciones de recuperación de imágenes.

- **Comprensión de Métricas Complejas:** La evaluación de un *pipeline* combinado de detección y clasificación requiere una comprensión profunda de métricas como Precision, Recall, F1-Score, IoU y mAP. La identificación del *recall* como el principal cuello de botella en nuestro *pipeline* es un ejemplo de cómo estas métricas guían el análisis y la mejora.
- **Valor de la Optimización de Modelos:** TFLite y la cuantización ofrecen una forma sencilla pero muy efectiva de preparar modelos para despliegue, equilibrando el rendimiento computacional con la precisión.

Desafíos y Soluciones

El desarrollo del proyecto no estuvo exento de desafíos:

1. **Desbalance Extremo del Dataset:**
    - **Desafío:** El dataset original presentaba un fuerte desbalance de clases, lo que podía llevar a un *overfitting* en clases mayoritarias y un bajo rendimiento en minoritarias.
    - **Abordaje:** Se implementó una estrategia de *data augmentation* para sobremuestrear las clases minoritarias, utilizando transformaciones como volteos, rotaciones, ajustes de contraste y brillo. Esto equilibró el dataset y mejoró la robustez del modelo.
2. **Recursos Computacionales:**
    - **Desafío:** El entrenamiento de modelos profundos como ResNet50 y la construcción de índices FAISS puede ser intensivo en memoria y CPU/GPU.
    - **Abordaje:** Se utilizó Google Colab, que proporciona acceso a GPUs, facilitando el entrenamiento y la manipulación de grandes volúmenes de datos.
3. **Ajuste de Hiperparámetros y Entrenamiento Robusto:**
    - **Desafío:** Encontrar la combinación óptima de tasa de aprendizaje, tamaño de lote y estrategias de regularización para evitar el *overfitting* y lograr una buena convergencia.
    - **Abordaje:** Se utilizaron *callbacks* de Keras como EarlyStopping para detener el entrenamiento cuando la validación deja de mejorar,
      y ReduceLROnPlateau para ajustar dinámicamente la tasa de aprendizaje, lo que automatiza gran parte de este proceso.
4. **Limitaciones de Detección de YOLOv8:**

- **Desafío:** Aunque YOLOv8 es rápido, su rendimiento de *recall* en imágenes complejas (con perros pequeños, ocluidos o en poses difíciles) no fue perfecto, lo que se convirtió en el principal cuello de botella del *pipeline* integrado.
- **Abordaje:** Se identificó claramente que el problema no era la clasificación de la raza (que funcionaba muy bien en los recortes), sino la detección inicial. Si bien no se realizó finetuning al modelo YOLO en este proyecto, la evaluación precisa de las métricas (Recall, mAP@0.5) permitió diagnosticar esta limitación y planificar futuras mejoras.

5. **Integración de Componentes Diversos:**
   - **Desafío:** Combinar modelos de diferentes frameworks (YOLO de Ultralytics, Keras/TensorFlow para clasificación) y FAISS para una experiencia fluida.
   - **Abordaje:** Se diseñaron clases modulares (DogSearchEngine, DogDetectionClassificationPipeline) para encapsular la lógica de cada componente y asegurar una interfaz limpia entre ellos. Gradio facilitó la creación de interfaces de usuario para probar y demostrar el sistema.

Aplicabilidad en la Vida Real

El sistema desarrollado tiene una **alta aplicabilidad en la vida real**:

- **Aplicaciones Móviles de Identificación de Razas:** Usuarios pueden tomar una foto de un perro y obtener instantáneamente su raza, útil para amantes de los perros, curiosos o futuros dueños.
- **Asistencia en Refugios de Animales:** Ayuda en la rápida identificación de razas para perros sin pedigrí conocido, facilitando la gestión y la adopción.
- **Medicina Veterinaria:** Soporte inicial en el diagnóstico y la comprensión de predisposiciones genéticas asociadas a ciertas razas.
- **Búsqueda de Mascotas Perdidas:** Un motor de búsqueda por similitud visual puede ayudar a encontrar mascotas con características similares.
- **Comercio Electrónico y Personalización:** Recomendar productos específicos para razas (alimento, juguetes, accesorios) basándose en la identificación.

Conclusión General sobre Utilidad: Los productos generados en este TP son de **alta utilidad práctica**. La combinación de una clasificación precisa con una detección en imágenes

complejas, sumado a la capacidad de búsqueda por similitud, crea una herramienta potente. La optimización con TFLite es fundamental para llevar esta funcionalidad a dispositivos móviles o *edge devices*, abriendo un abanico de posibilidades para el despliegue.

Posibilidades de Aplicación y Calidad Obtenida: La calidad obtenida, especialmente el 88% de precisión en clasificación y el alto IoU en detección, es muy prometedora. Sin embargo, el *recall* del *pipeline* (0.6190) es un área clave para mejorar si la aplicación requiere una detección extremadamente exhaustiva.

Referencias

- **Dataset:**
  - Senka, G. (2020). *70-Dog-Breeds-Image-Data-Set*. Kaggle.
- **Redes Neuronales Convolucionales y Transfer Learning:**
  - He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. (ResNet50)
  - Chollet, F. (2015). Keras: The Python Deep Learning library. https://keras.io/
  - Abadi, M., et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.
- **Bases de Datos Vectoriales y Búsqueda por Similitud:**
  - Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*. (Para FAISS)
  - FAISS GitHub Repository
- **Detección de Objetos (YOLO):**
  - Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.
  - Ultralytics YOLOv8 Documentation
- **Optimización de Modelos (TFLite):**
  - TensorFlow Lite Documentation
- **Librerías y Herramientas:**

- OpenCV Documentation
- NumPy Documentation
- Pandas Documentation
- Matplotlib Documentation
- Scikit-learn Documentation
- Gradio Documentation