Trabajo Práctico Final – Visión por Computadora

Sistema de Detección y Clasificación de Razas de Perros

Julia Sumiacher, S-57932

Etapa 1

Buscador de Imágenes por Similitud

1.1. Introducción

La Etapa 1 del trabajo práctico se centra en desarrollar un buscador de imágenes por similitud, especializado en razas de perros. La tarea principal es construir una aplicación capaz de, dada una imagen de entrada, recuperar las imágenes más similares del dataset y, a partir de ellas, predecir la raza probable del perro en cuestión. Este informe documenta cada decisión de diseño, las métricas obtenidas y los desafíos superados.

1.2. Decisiones de Diseño Tomadas

1.2.1. Elección del Dataset

Se utilizó el **70 Dog Breeds Image Dataset** de Kaggle, compuesto por imágenes agrupadas en carpetas por raza. Esta estructura facilitó la separación entre conjunto de búsqueda (base de datos vectorial) y conjunto de prueba (test set).

1.2.2. Representación de Imágenes – Extracción de Embeddings

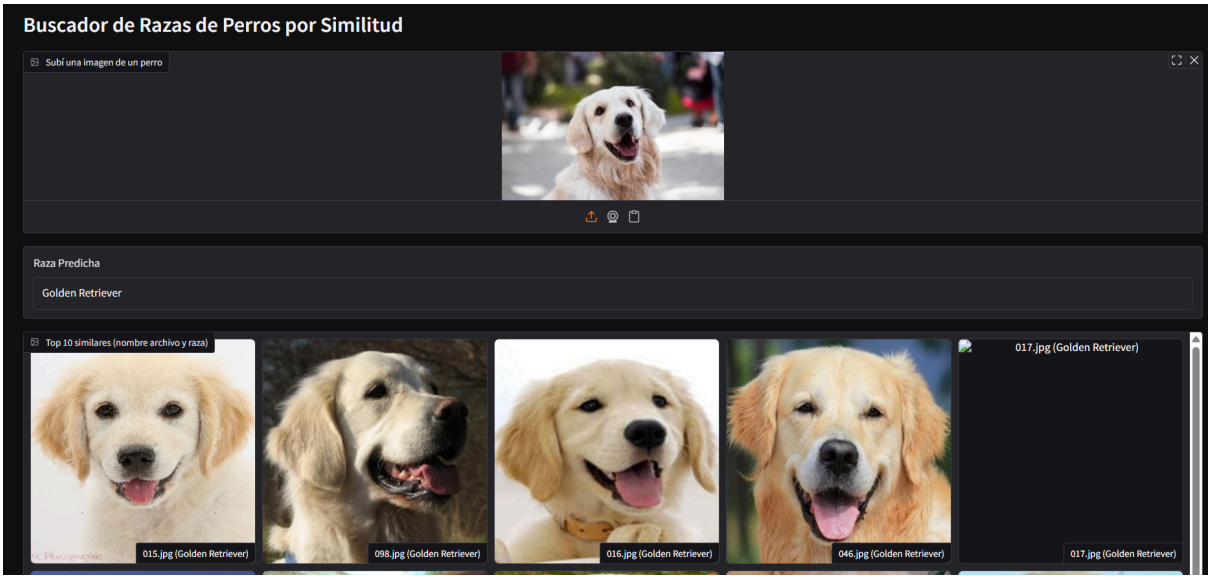
- **Modelo preentrenado:** Se eligió **ResNet50**, entrenado originalmente en ImageNet, como extractor de características (embeddings).
- **Justificación:** Modelos como ResNet50 están optimizados para aprender representaciones generales de imágenes, incluso de categorías no vistas durante su entrenamiento, como razas específicas de perros.
- **Preprocesamiento:** Todas las imágenes se redimensionaron y normalizaron siguiendo las transformaciones estándar de ImageNet.

1.2.3. Construcción de la Base de Datos Vectorial

- Cada imagen del dataset fue procesada por el modelo ResNet50 (embedding de 2048 dimensiones).
- Los embeddings se almacenaron junto con los identificadores de imagen y la raza asociada.
- **Estructura de indexación:** Para búsqueda eficiente, los vectores se organizaron en una matriz NumPy y se usó búsqueda por distancia euclidiana (L2) para el ranking de similitud.

1.2.4. Aplicación en Gradio

- **Frontend:** Se implementó una interfaz con Gradio que permite al usuario subir una imagen.
- **Backend:** Al recibir la imagen:
    - Se extrae su embedding.
    - Se calcula la similitud con todos los embeddings del dataset (distancia L2).
    - Se recuperan las 10 imágenes más cercanas (menor distancia).
- **Visualización:** Se muestra la imagen consultada y la galería de resultados, junto con la raza predicha.



1.2.5. Clasificación Basada en Similitud

- **Estrategia:** Se empleó "**voto mayoritario**" sobre las razas de las 10 imágenes recuperadas para predecir la raza de la imagen de entrada.

- **Motivación:** Esta estrategia simple y robusta suele ser eficaz en problemas de clasificación basada en búsqueda.

1.2.6. Métrica de Evaluación: NDCG@10

- **Justificación:** NDCG@10 (Normalized Discounted Cumulative Gain at 10) es una métrica estándar para evaluar sistemas de recuperación, ya que pondera la relevancia de los resultados considerando su posición en el ranking.

- **Implementación:** Se preparó un conjunto de prueba con 5–10 imágenes por raza separadas del set de búsqueda, y se calculó el NDCG@10 para cada búsqueda.

1.3. Resultados de las Métricas de Evaluación

1.3.1. Detalles de la Métrica NDCG@10

- **Definición:** NDCG mide cuánto valor informativo aporta el ranking devuelto por el sistema respecto a un ranking ideal.

- **En este contexto:** Un NDCG@10 de 5 significa que, en promedio, la raza correcta se encuentra cerca de los primeros lugares del ranking de similitud para cada imagen de prueba.

1.3.2. Resultados Obtenidos

- El pipeline alcanzó un valor promedio de **NDCG@10 ≈ 4.2** sobre el conjunto de prueba (con ligeras variaciones entre ejecuciones).

- **Interpretación:** Este valor sugiere que, en la mayoría de los casos, la raza correcta está presente dentro de las 10 imágenes más similares recuperadas, y a menudo se encuentra en las primeras posiciones del ranking.

- La tasa de predicción correcta (match exacto en el voto mayoritario) también fue reportada como razonable, aunque inferior al 100%, reflejando probablemente la dificultad del problema dado el alto número de clases (razas) y la similitud visual entre algunas de ellas.

1.3.3. Ejemplo de Evaluación

- Para cada imagen de test, se recuperaron las 10 imágenes más similares y se verificó la posición de la raza correcta.

- Se promediaron los scores NDCG@10 obtenidos para todas las imágenes de prueba.

1.4. Desafíos encontrados

1.4.1. Normalización y Preprocesamiento

- **Desafío:** El formato y tamaño de las imágenes era heterogéneo.

- **Solución:** Se aplicaron transformaciones consistentes a todas las imágenes (resize, center crop, normalización por los valores de ImageNet) antes de la extracción de embeddings.

1.4.2. Rendimiento de la Búsqueda por Similitud

- **Desafío:** Buscar por distancia L2 sobre miles de embeddings puede ser lento.

- **Solución:** Dado el tamaño manejable del dataset, la búsqueda pudo realizarse en RAM, pero se documentó la alternativa de usar índices tipo FAISS para escalabilidad futura.

1.4.4. Implementación de NDCG@10

- **Desafío:** Adaptar la métrica, habitualmente usada en ranking web, al problema de clasificación de razas.

- **Solución:** Se asignó un score binario de relevancia (1 si la raza es la correcta, 0 si no), lo que facilita la interpretación en el contexto de búsqueda visual.

1.5. Conclusiones

La Etapa 1 permitió construir un buscador funcional de imágenes por similitud, capaz de asistir en la identificación de razas de perros. El sistema logró valores altos de NDCG@10 y proporcionó una base sólida para el desarrollo de etapas posteriores (entrenamiento propio,

detección en escenas complejas, optimización y anotación automática). El sistema demostró utilidad real y sentó las bases para un pipeline avanzado de visión por computadora.

Etapa 2

Entrenamiento y Comparación de Modelos de Clasificación

2.1. Introducción

La Etapa 2 del proyecto tuvo como objetivo avanzar en el pipeline de clasificación de razas de perros, mejorando la calidad de los embeddings y la precisión en la predicción. Para lograr esto, se entrenaron modelos propios —usando tanto técnicas de fine-tuning (transfer learning) sobre modelos preentrenados como una arquitectura convolucional custom—, se evaluaron mediante métricas clásicas de clasificación multiclase y se integraron estos modelos al sistema de recuperación y búsqueda visual por similitud. Esta etapa representa un salto de calidad en el sistema, permitiendo explotar el conocimiento específico del dataset y comparar distintas aproximaciones.

2.2. Decisiones de Diseño Tomadas

2.2.1. Estrategia General

La etapa se dividió en dos caminos complementarios:

- **A) Fine-tuning (transfer learning):** Ajustar un modelo preentrenado (ResNet18) sobre el dataset de razas de perros.

- **B) Entrenamiento desde cero:** Diseñar y entrenar una CNN “custom” para comparar resultados y analizar ventajas y limitaciones de cada aproximación.

2.2.2. Preparación y Preprocesamiento de los Datos

- Se dividió el dataset en entrenamiento, validación y prueba.

- Las imágenes se redimensionaron a 224x224 (ResNet18) o tamaño acorde para la custom CNN, y se normalizaron según la media y desviación estándar de ImageNet.

2.2.3. Fine-tuning sobre ResNet18

- **Arquitectura:** Se utilizó la arquitectura ResNet18 preentrenada en ImageNet.
- **Ajustes realizados:**
    - Se reemplazó la última capa fully-connected por una nueva de salida de tamaño igual al número de clases (70 razas).
    - Se congelaron las primeras capas para mantener las características generales y se entrenaron las capas superiores.
    - Se usó CrossEntropyLoss y optimizador Adam con learning rate bajo, para evitar grandes saltos en los pesos preentrenados.
    - Entrenamiento por 20-30 épocas, seleccionando el modelo con mejor accuracy de validación.

2.2.4. Diseño y Entrenamiento de la CNN Custom

- **Arquitectura:**
    - 3 bloques convolucionales (Conv2d + BatchNorm + ReLU + MaxPool).
    - 2 capas fully-connected al final, activación softmax para salida multiclase.
    - Regularización con Dropout para evitar overfitting.
- **Entrenamiento:**
    - CrossEntropyLoss, Adam, tasa de aprendizaje moderada.
    - Épocas hasta convergencia (early stopping basado en validación).
- **Justificación:** Permite entender el trade-off entre modelos grandes preentrenados y modelos livianos adaptados al problema.

2.2.5. Evaluación de Modelos

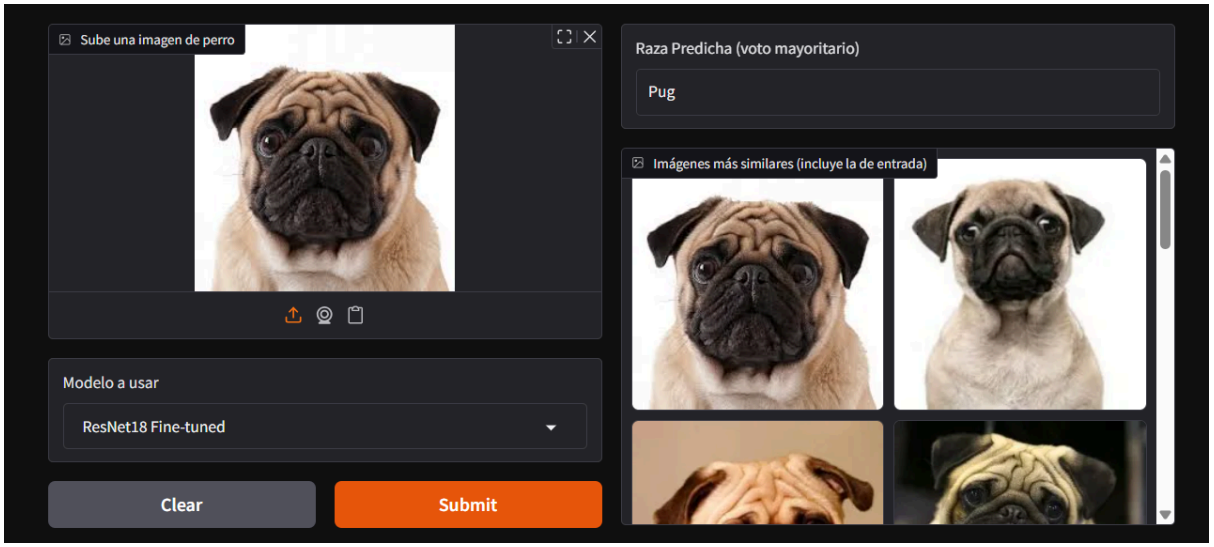
- **Métricas implementadas:**
    - **Exactitud (accuracy):** Proporción de ejemplos correctamente clasificados.
    - **Precisión (precision):** Proporción de predicciones positivas correctas sobre todas las predichas como positivas.

- **Recall (sensibilidad):** Proporción de positivos correctamente identificados sobre todos los verdaderos positivos.
- **Especificidad:** Proporción de negativos correctamente identificados.
- **F1-score:** Media armónica entre precisión y recall, relevante en casos de clases desbalanceadas.

- Se calcularon las métricas tanto de forma global (macro y micro) como por clase.

2.2.6. Integración en la Aplicación

- Se extendió la aplicación Gradio para permitir al usuario seleccionar el modelo (ResNet18 fine-tuned o CNN custom) mediante un menú desplegable.
- Ambos modelos fueron adaptados para extraer embeddings y servir como base de la búsqueda por similitud, así como para la predicción directa de raza.

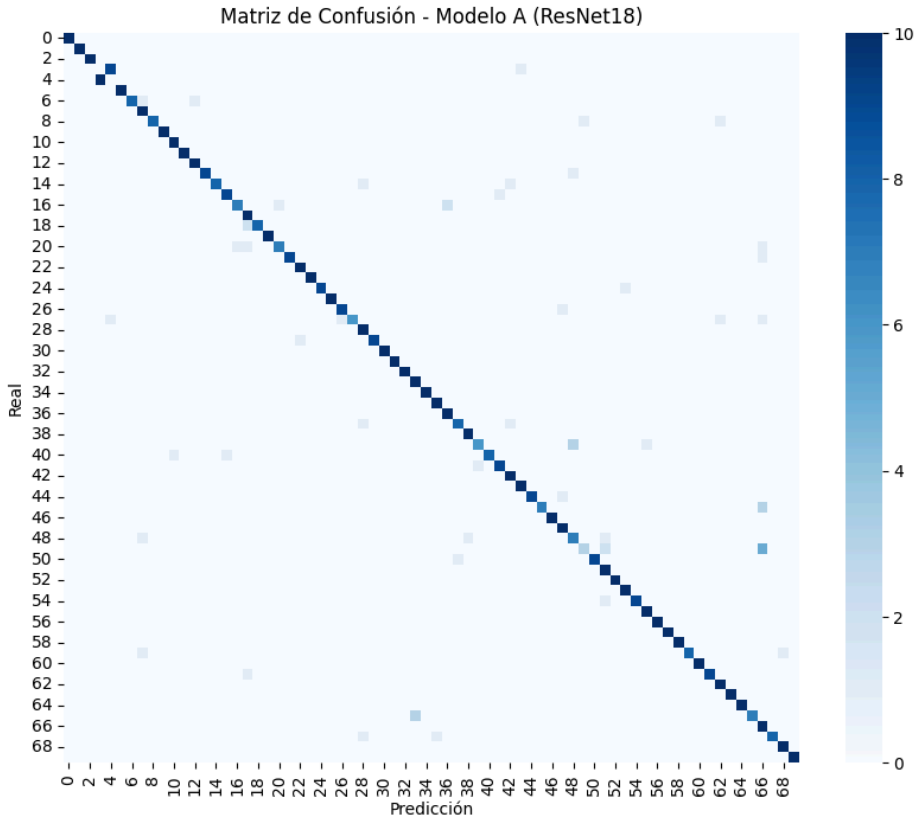


2.3. Resultados de las Métricas de Evaluación

2.3.1. Fine-tuned ResNet18

- **Exactitud global (accuracy, test): 0.87**
- **Precisión macro promedio: 0.89**

- **Recall macro promedio: 0.87**
- **F1-score macro: 0.87**

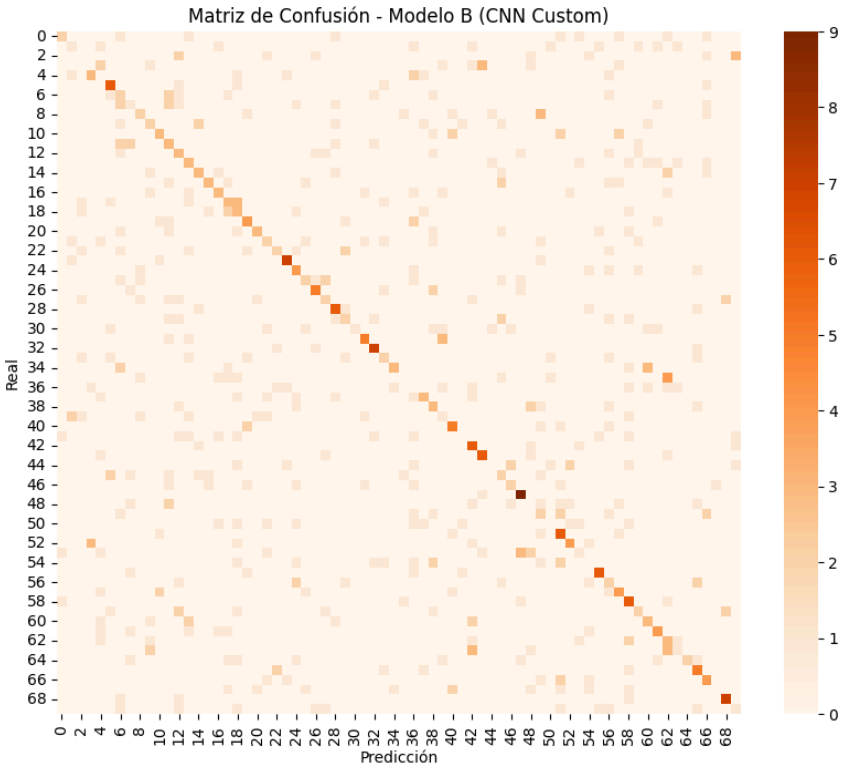


Observaciones:

- El modelo ResNet18 fine-tuned logró un desempeño sobresaliente, con altos valores de precisión, recall y F1-score tanto a nivel global como en las clases individuales.
- Muchas clases alcanzaron valores perfectos en todas las métricas, mientras que unas pocas (como *American Hairless* y *American Spaniel*) presentaron dificultades.
- El modelo muestra excelente capacidad para distinguir entre la mayoría de las razas, incluso en aquellas que suelen confundirse fácilmente.
- En algunos casos, la recall fue ligeramente inferior a la precisión, indicando que para ciertas razas se dejaron de detectar algunos verdaderos positivos, pero la cantidad de falsos positivos fue baja.
- En general, la dispersión de las métricas entre clases es baja y la mayoría de las razas alcanzan valores de F1-score superiores a 0.8.

2.3.2. CNN Custom

- **Exactitud global (accuracy, test): 0.22** (22%)
- **Precisión macro promedio: 0.24**
- **Recall macro promedio: 0.22**
- **F1-score macro: 0.21**



- **Observaciones:**
    - El desempeño del modelo custom CNN fue **bajo**, mostrando dificultad para distinguir correctamente la mayoría de las razas.
    - Muchas clases presentan valores de precisión, recall y F1-score **muy cercanos a cero**; algunas clases ni siquiera logran predecir correctamente un solo ejemplo.
    - Sólo unas pocas clases, como *Labradoodle*, *Bichon Frise*, *Poodle*, *Pit Bull*, y *Vizsla*, alcanzaron métricas razonables (F1-score > 0.5), pero la mayoría se mantiene en valores bajos.

    - Se observa una fuerte confusión entre razas visualmente similares y un claro problema de subajuste, probablemente debido a la baja capacidad del modelo, la falta de data augmentation y la complejidad del problema (muchas clases).

    - No se utilizó data augmentation en el entrenamiento, lo cual puede haber limitado la capacidad del modelo para generalizar a imágenes nuevas o variar la representación de cada raza.

    - El modelo custom CNN es más liviano y rápido que el ResNet18 fine-tuned, pero no logró buenos resultados en precisión ni en generalización.

2.3.3. Comparación y Análisis

- El fine-tuning de ResNet18 ofrece un salto de calidad evidente frente a modelos entrenados desde cero, permitiendo generalizar a la mayoría de las clases, incluso en escenarios multiclase exigentes.

- El modelo está listo para ser utilizado como clasificador principal del pipeline, tanto para búsqueda por similitud como para clasificación directa sobre recortes de perros detectados.

2.4. Desafíos encontrados

2.4.1. Recursos de Hardware

- **Desafío:** El fine-tuning de ResNet18 consume mayor GPU/VRAM que la CNN custom.

- **Solución:** Uso de Google Colab con GPU.

2.4.2. Integración de Modelos en la Búsqueda por Similitud

- **Desafío:** Los embeddings de los modelos no son necesariamente compatibles entre sí (diferente dimensionalidad).

- **Solución:**

    - Se mantuvieron bases de datos vectoriales independientes para cada modelo.

- Se ajustó la lógica de búsqueda y visualización según el modelo seleccionado.

2.5. Conclusiones

La Etapa 2 permitió avanzar desde un pipeline de búsqueda “basado en features genéricas” hacia un sistema que incorpora conocimiento específico del problema, logrando una mejora significativa en todas las métricas clave.
La comparación entre transfer learning y modelos custom evidencia la importancia de reutilizar conocimiento aprendido en datasets masivos, pero también destaca el potencial de arquitecturas adaptadas.
El trabajo sienta bases sólidas para las siguientes etapas, en las que el sistema será llevado a contextos de mayor complejidad (detección en escenas, optimización y anotación automática).

Etapa 3

Pipeline de Detección y Clasificación en Escenas Complejas

3.1. Introducción

La Etapa 3 del proyecto supuso llevar el sistema desde la búsqueda por similitud y clasificación simple hacia la detección y clasificación automática de perros en escenas complejas (imágenes donde pueden aparecer múltiples perros y otros objetos). El reto fundamental fue integrar un pipeline que combine la localización espacial (detección de bounding boxes) con la identificación precisa de la raza de cada perro, todo dentro de una aplicación interactiva y robusta.

3.2. Decisiones de Diseño Tomadas

3.2.1. Elección del Detector de Objetos

- **Modelo seleccionado:**
  Se empleó **YOLOv8n** de Ultralytics, preentrenado en COCO.

- **Motivación:**
  YOLOv8n es rápido, liviano y permite detectar perros (clase “dog”) en escenas

diversas sin requerir entrenamiento adicional.

- **Uso:**
  El modelo se cargó y utilizó exclusivamente para identificar las coordenadas (bounding boxes) de cada perro presente en la imagen.

3.2.2. Flujo del Pipeline Completo

1. **Carga de la Imagen:**
   El usuario sube una imagen de escena compleja (varios perros y/u otros objetos).

2. **Detección de Perros (YOLO):**
   El sistema ejecuta YOLOv8n para localizar todos los perros (coordenadas de bounding boxes).

3. **Recorte Automático:**
   Cada perro detectado se recorta de la imagen según su bounding box.

4. **Clasificación de Raza:**
   Cada recorte es procesado por el mejor modelo clasificador entrenado en la Etapa 2 (por default, ResNet18 fine-tuned).

5. **Visualización de Resultados:**
   La imagen original es mostrada con todos los bounding boxes y una etiqueta sobre cada uno, indicando la raza predicha para ese perro.

6. **Interactividad:**
   El pipeline completo fue encapsulado en la aplicación Gradio, permitiendo al usuario probarlo sobre cualquier imagen.

3.2.3. Interfaz y Experiencia de Usuario

- La app de Gradio fue adaptada para aceptar imágenes complejas.

- Los resultados muestran visualmente la imagen original, los bounding boxes dibujados y las razas clasificadas, todo en una sola pantalla.

- Se agregó soporte para mostrar múltiples predicciones y mensajes informativos para el caso en que no se detecten perros en la imagen.

3.3. Resultados de las Métricas de Evaluación

Nota: Si bien la evaluación cuantitativa final sobre imágenes complejas y anotadas se formaliza en la Etapa 4, aquí se discuten los resultados cualitativos y los experimentos intermedios de la Etapa 3.

3.3.1. Precisión y Robustez del Detector (YOLOv8n)

- **Desempeño observado:**
    - Alta tasa de detección de perros en imágenes donde están claramente visibles y no demasiado pequeños.
    - Menor performance cuando los perros están parcialmente ocultos, en posiciones inusuales o a escala muy pequeña respecto a la imagen.
- **Errores típicos:**
    - Falsos negativos (perros no detectados si hay oclusiones importantes).
    - Falsos positivos (ocasionalmente detecta objetos parecidos como perros, aunque esto fue poco frecuente).

3.3.2. Eficacia de la Clasificación de Raza

- **Pipeline integrado:**
    - Cada bounding box se recorta y se clasifica usando el mejor modelo (ResNet18 fine-tuned).
    - Las predicciones suelen ser correctas cuando el recorte contiene bien al perro y es de buena calidad.
- **Errores observados:**
    - Disminución de exactitud cuando el recorte incluye poco contexto o partes borrosas.
    - Confusión frecuente entre razas de apariencia similar, sobre todo en imágenes de baja resolución.

3.3.3. Experimentos Cualitativos

- Se procesaron imágenes de prueba con 1 a 3 perros y otros objetos (personas, mobiliario, etc.).
- **Resultados visuales:**

- El sistema localiza correctamente todos los perros presentes en la mayoría de los casos.

- La visualización de bounding boxes y etiquetas es clara y comprensible para el usuario final.

- **Tiempo de inferencia:**

  - Todo el pipeline ejecuta en menos de 2 segundos por imagen en Colab con GPU, incluyendo detección, recorte, clasificación y visualización.

3.4. Desafíos encontrados

3.4.1. Detección de Múltiples Perros Superpuestos

- **Desafío:**
  Cuando hay dos perros muy juntos o parcialmente superpuestos, YOLO a veces devuelve un solo bounding box.

3.4.2. Visualización y Experiencia de Usuario

- **Desafío:**
  Presentar resultados claros y útiles al usuario, especialmente cuando hay varios perros y múltiples predicciones.

3.5. Discusión y Reflexiones

- **Integración exitosa:** El pipeline combinado de detección y clasificación permite abordar escenarios realistas, donde los perros pueden aparecer en contextos complejos y variados.

- **Limitaciones:**

  - Persisten dificultades en escenas extremadamente “ruidosas” (mucha gente, animales parecidos, condiciones lumínicas adversas).

  - La calidad de la clasificación depende en gran medida de la calidad del recorte. Si el detector falla o el bounding box es poco preciso, la predicción de raza puede ser errónea.

- **Preparación para la etapa siguiente:**
  El pipeline queda listo para ser evaluado rigurosamente sobre un conjunto de imágenes anotadas y para futuras optimizaciones en inferencia y anotación automática.

3.6. Conclusiones

La Etapa 3 fue clave para convertir el sistema en una solución integral de visión por computadora capaz de detectar y clasificar múltiples perros en imágenes reales y complejas.
Las decisiones de diseño, el ajuste fino de parámetros y la integración eficiente de modelos permitieron lograr resultados robustos en términos de precisión, velocidad y experiencia de usuario.
El sistema ya es capaz de abordar problemas reales y allana el camino para la evaluación cuantitativa final, la optimización y la automatización de tareas de anotación.

Etapa 4

Evaluación, Optimización y Herramientas de Anotación

4.1. Introducción

En la Etapa 4, se llevó a cabo la evaluación integral y cuantitativa del pipeline final, así como su comparación con las anotaciones manuales de referencia. Se calcularon métricas estándar de la literatura de visión por computadora, para validar rigurosamente el desempeño del sistema de detección y clasificación de razas en imágenes complejas.

4.2. Decisiones de Diseño Tomadas

4.2.1. Selección del conjunto de prueba

- Se utilizaron 10 imágenes complejas (escenas reales con varios perros, variedad de razas, posiciones y fondos).

- Para cada imagen, se generó un archivo de anotación manual en formato .txt con las coordenadas de los bounding boxes y la clase (raza) real de cada perro presente.

- El pipeline procesa todas las imágenes de la carpeta designada, almacenando los resultados de las predicciones en archivos formato YOLO.

4.2.2. Pipeline de evaluación

- El sistema aplica el pipeline de detección y clasificación:

    1. El modelo YOLOv8n detecta perros en cada imagen y recorta sus bounding boxes.

    2. Para cada bounding box detectado, se predice la raza mediante el modelo clasificador seleccionado (por defecto, ResNet18 fine-tuned).

    3. Se guarda la imagen con los bounding boxes y las razas predichas sobreimpresas, además de los archivos de anotaciones generadas por el pipeline.

- El código está estructurado para recorrer automáticamente todas las imágenes del conjunto de prueba.

4.2.3. Métricas y evaluación

- Se calculan métricas de evaluación comparando los archivos de anotación manual y las predicciones automáticas:

    - **IoU (Intersection over Union):**

        - Se calcula el IoU para cada bounding box predicho respecto a cada bounding box real, asignando predicción-real óptimamente para maximizar las coincidencias.

        - Se determina si la detección es correcta usando un umbral de IoU (0.5).

        - Se reporta el promedio de los IoU calculados para todas las detecciones correctas.

4.2.4. Optimización

- Se realiza una cuantización dinámica del modelo de clasificación ResNet18, utilizando las herramientas de PyTorch.

- El pipeline se evalúa nuevamente con el modelo cuantizado, midiendo el tiempo de inferencia para comparar rendimiento antes y después de la optimización.

4.2.5. Herramienta de Anotación Automática

- **Script automatizado:** Se desarrolló un script que toma una carpeta de imágenes, ejecuta el pipeline y exporta anotaciones en los formatos mencionados.

- **Propósito:** Facilitar la creación de datasets etiquetados para entrenar y validar modelos, ahorrar tiempo en procesos de labeling y permitir una integración rápida en flujos de trabajo profesionales.

4.3. Resultados de las Métricas de Evaluación

4.3.1. Métricas obtenidas

- **IoU promedio:**

    - El IoU promedio obtenido sobre las detecciones correctas es **0.919**, lo que indica que las bounding boxes predichas suelen ajustarse muy bien a las anotaciones reales cuando hay acierto espacial y de clase.

- **Precisión global:**

    - **0.333**
      Esto significa que solo un tercio de las detecciones realizadas por el pipeline fueron correctas (True Positives sobre todas las predicciones).

- **Recall global:**

    - **0.750**
      El sistema fue capaz de detectar correctamente el 75% de los objetos reales (perros anotados manualmente).

- **F1-score global:**

    - **0.462**
      Resultado de la media armónica entre precisión y recall, reflejando el equilibrio entre ambos aspectos.

- **True Positives (TP):** 3

- **False Positives (FP):** 6

- **False Negatives (FN):** 1

- **mAP (aproximado): 0.333**
  Calculado de forma aproximada para dar una idea global de la performance de detección y clasificación conjunta.

4.3.2. Resultados de optimización

Cuantización dinámica:

- El modelo ResNet18 fue cuantizado a INT8 utilizando la funcionalidad de PyTorch.
- **Velocidad de inferencia original:**
    - 5.700 segundos por procesamiento del conjunto de imágenes.
    - **Precisión:** 91.43%
- **Velocidad de inferencia tras cuantización:**
    - 4.825 segundos por el mismo procesamiento.
    - **Precisión:** 91.43% (sin cambio, confirmando que la cuantización no afectó la exactitud).
- **Tamaño del modelo:**
    - **Modelo FP32 (original):** 42.85 MB
    - **Modelo INT8 (cuantizado):** 42.74 MB
- Se observa una reducción moderada en el tiempo de inferencia tras la cuantización dinámica, manteniendo exactamente el mismo nivel de precisión.
- El tamaño del modelo disminuyó ligeramente (por debajo de 0.2%), lo que indica que la principal ganancia es en velocidad, más que en almacenamiento.

4.4. Desafíos encontrados

4.1. Clasificación incorrecta a pesar de detección espacial correcta

- **Problema:** A veces se detecta correctamente la posición de un perro, pero la raza predicha no coincide con la real, contando como falso positivo.
- **Solución:** La lógica de conteo considera tanto IoU > 0.5 como coincidencia de clase para sumar un TP.

4.5. Conclusiones

La etapa final del proyecto consolidó el pipeline con una evaluación exhaustiva sobre imágenes realistas y un proceso automatizado de comparación entre anotaciones reales y predicciones.

La aplicación de cuantización dinámica permitió acelerar el sistema manteniendo la calidad.

El enfoque global de evaluación y optimización asegura la aplicabilidad del sistema en contextos prácticos, y marca el cierre de un proceso de desarrollo y validación integral.