Tecnicatura Universitaria en Inteligencia Artificial

Trabajo Práctico Final Visión por Computadora

Autor: Caballero Franco

Asignatura: Visión por Computadora

Profesores: Juan Pablo Manson y Constantino Ferrucci

Fecha de entrega: 27/11/2025

ÍNDICE

RESUMEN

Este trabajo práctico desarrolla la fase inicial de un **pipeline de visión por computadora** para la identificación de 70 razas de perros a partir de imágenes, centrándose en el **análisis de similitud** y la **clasificación por *Deep Learning***

El objetivo inicial (Etapa 1) fue crear un **Buscador de Imágenes por Similitud**. Para esto, se utilizó un modelo **ResNet50 pre-entrenado en ImageNet** para extraer embeddings (vectores de características) de cada imagen del dataset. Estos vectores se indexaron en una base de datos vectorial para permitir búsquedas eficientes. Se desarrolló una aplicación en **Gradio** que, al recibir una imagen de entrada, la procesa, busca las 10 imágenes más similares por distancia vectorial y predice la raza mediante el voto mayoritario de los resultados recuperados. La calidad del sistema de búsqueda se evaluó utilizando la métrica **NDCG@10**.

La segunda etapa (Etapa 2) se enfocó en el **entrenamiento de modelos personalizados** para mejorar la representación de características y la precisión de la clasificación. Se realizó el fine-tuning de un modelo **ResNet18** (Modelo A) sobre el dataset de 70 razas. Posteriormente, se entrenó un modelo **CNN custom** (Modelo B) desde cero. La aplicación Gradio fue actualizada para permitir al usuario seleccionar qué modelo (ResNet50 de ImageNet, ResNet18 fine-tuned o CNN custom) desea utilizar para generar los embeddings de búsqueda por similitud.

INTRODUCCIÓN

La **identificación automática de razas de perros** en imágenes es un desafío fundamental en el campo de la **Visión por Computadora (Computer Vision)** con diversas aplicaciones prácticas, que van desde la gestión de refugios de animales y la localización de mascotas perdidas hasta el desarrollo de herramientas veterinarias. La dificultad de esta tarea radica en la **alta variabilidad intra-raza** (diferentes poses, iluminación, edad) y la **baja variabilidad inter-raza** (razas muy similares morfológicamente).

Este trabajo se justifica en la necesidad de desarrollar un sistema robusto y eficiente que no solo clasifique imágenes limpias, sino que también funcione en **escenas complejas** del mundo real (lo que se abordará en etapas posteriores). Las primeras etapas del proyecto (cubiertas en este informe parcial) se centran en establecer una **base de conocimiento visual** mediante dos pilares tecnológicos:

- **Sistemas de Recuperación de Imágenes (Image Retrieval):** Utilizar **embeddings** y bases de datos vectoriales para encontrar similitudes visuales rápidamente, permitiendo una clasificación inicial basada en el **conocimiento por pares (k-Nearest Neighbors)**.
- **Modelos de *Deep Learning* Específicos:** Entrenar y optimizar modelos convolucionales (CNNs) mediante **Transfer Learning** para aprender representaciones de características altamente discriminatorias que superen la generalidad de los modelos pre-entrenados como ResNet50 en ImageNet.

El desarrollo de este *pipeline* completo de *Computer Vision* sirve como un ejercicio integral de ingeniería de *Machine Learning*, abarcando desde la extracción de características hasta la optimización del modelo y la integración en una aplicación interactiva.

El alcance de este informe parcial cubre las Etapas 1 y 2, y sus objetivos específicos fueron:

- **Desarrollar un Buscador de Imágenes por Similitud:** Implementar la extracción de *embeddings* utilizando un modelo pre-entrenado (ResNet50) y construir una **base de datos vectorial** para realizar búsquedas eficientes de las 10 imágenes más similares a una imagen de consulta.
- **Implementar Clasificación Basada en Similitud:** Crear una lógica de **"voto mayoritario"** a partir de los resultados de la búsqueda para predecir la raza de una imagen de entrada.
- **Evaluar la Calidad de la Búsqueda:** Medir el rendimiento del sistema de recuperación de imágenes utilizando la métrica **NDCG@10 (Normalized Discounted Cumulative Gain)**.
- **Entrenar y Comparar Modelos de Clasificación:** Realizar el ***fine-tuning*** de un modelo **ResNet18** y entrenar un modelo **CNN *custom*** sobre el *dataset* de 70 razas de perros para obtener representaciones de características mejoradas.
- **Integrar Modelos en la Aplicación:** Actualizar la interfaz de **Gradio** para permitir al usuario seleccionar dinámicamente el modelo (ResNet50, ResNet18 *fine-tuned* o CNN *custom*) que se utilizará para generar los *embeddings* de búsqueda.

A continuación, se presenta la estructura del informe para facilitar la comprensión del desarrollo y los resultados obtenidos.

Estructura del Informe

La estructura del informe se organiza en las siguientes secciones: metodología, donde se detallan las decisiones técnicas y herramientas utilizadas; desarrollo e implementación, donde se describe el proceso de construcción del sistema; resultados, donde se presentan las pruebas realizadas; y finalmente, las conclusiones, que incluyen un análisis crítico del desempeño y posibles mejoras futuras.

METODOLOGÍA

Fuente de datos:

El proyecto se desarrolló utilizando un *dataset* público de alta calidad, cuyas características se detallan a continuación:

- **Fuente: 70 Dog Breeds Image Dataset** (plataforma Kaggle). Link Dataset
- **Contenido:** El *dataset* contiene imágenes asociadas a **70 razas de perros** diferentes, lo que representa un desafío de **clasificación multi-clase** con un alto número de categorías.

- **Estructura:** Las imágenes están organizadas en carpetas, divididas en conjuntos de train, test y validación, donde dentro, cada carpeta corresponde al nombre de una raza específica, facilitando la implementación de técnicas de aprendizaje supervisado.

Herramientas y Tecnologías usadas:

El desarrollo del sistema se realizó utilizando el lenguaje de programación Python, utilizando el entorno de desarrollo Google Colab, complementado por las siguientes tecnologías y librerías clave.

- **TensorFlow:** Framework principal utilizado para la definición, entrenamiento (*fine-tuning*) y evaluación de los modelos de *Deep Learning* (ResNet18, ResNet50, CNN *Custom*).
- **Torch:** Librería de PyTorch para el manejo de tensores y algunas funcionalidades complementarias de DL/ML. (Específicamente usado para el modelo Custom)
- **FAISS:** Librería eficiente de Facebook AI Research para la creación y gestión del índice de la Base de Datos Vectorial, permitiendo búsquedas rápidas de vecinos más cercanos (*k*-NN).
- **Gradio:** Herramienta utilizada para construir la aplicación interactiva web del Buscador por Similitud, facilitando la interacción del usuario con el modelo.
- **Pandas:** Utilizado para el análisis exploratorio de datos, gestión de etiquetas y organización de metadatos del *dataset*.
- **Os:** Utilizado para interactuar con el sistema operativo, gestionar directorios y trabajar con la estructura de archivos e imágenes del *dataset*.
- **Matplotlib:** Empleado para la realización de gráficos de las curvas de entrenamiento, matrices de confusión y visualización de resultados de métricas.
- **Random:** Utilizado específicamente para la aplicación de técnicas de balanceo de clases, como el undersampling, durante la preparación del conjunto de entrenamiento.

Técnicas usadas:

Preprocesamiento

El análisis exploratorio del conjunto de entrenamiento reveló un significativo desbalance de clases, lo cual puede introducir un sesgo en el modelo de *Machine Learning*, favoreciendo la clasificación de clases mayoritarias. Se observó lo siguiente:

- **Varianza Extrema:** Se observó una amplia disparidad en la cantidad de muestras por clase (raza). Por ejemplo, la raza **Shin-Tzu** era la clase dominante con casi 200 imágenes, mientras que la raza **American Hairless** era la menos representada con alrededor de 70 imágenes.
- **Riesgo de Sesgo:** Esta asimetría implica que el modelo podría tener un rendimiento deficiente en la identificación de las clases minoritarias, resultando en un modelo poco robusto que no generaliza bien a las 70 razas.

Para corregir este sesgo, se decidió rebalancear el dataset utilizando la media de las imágenes por clase, que se sitúa en 113 imágenes, como punto de referencia:

- **Clases Sobre-representadas (Sobre Promedio):** Para las clases que superan las 113 imágenes, se aplicó **Undersampling** utilizando la librería *random.* Este proceso redujo la cantidad de imágenes utilizadas de estas clases mayoritarias hasta que su conteo se acercó al promedio.
- **Clases Sub-representadas (Bajo promedio):** Para las clases que tenían menos de 113 imágenes, se aplicó una técnica de **Oversampling** mediante **Data Augmentation**. Esto se implementó directamente en el *pipeline* de entrenamiento, generando nuevas variaciones sintéticas de las imágenes existentes hasta alcanzar el umbral promedio.

También se detectó y corrigió un error en el etiquetado del *dataset*. Específicamente, **se modificó el nombre de la carpeta** correspondiente a una de las razas que contenía una **nomenclatura incorrecta**. Este paso de limpieza fue crucial para garantizar que las etiquetas de clase (*ground truth*) fueran consistentes y evitar errores de mapeo en el posterior entrenamiento y evaluación de los modelos.

Decisiones Clave de Diseño:

Base de Datos Vectorial

La elección de la librería FAISS (Facebook AI Similarity Search) como base de datos vectorial para la Etapa 1 se justifica por su eficiencia computacional y su capacidad para manejar búsquedas en espacios de alta dimensión

- **Velocidad:** FAISS permite la implementación de índices de búsqueda de vecinos más cercanos (k-NN) optimizados, como **IndexIVF (Inverted File)**, que aceleran significativamente el tiempo de consulta en comparación con una búsqueda lineal bruta, algo vital para una aplicación interactiva como la construida con Gradio.
- **Escalabilidad:** Aunque el *dataset* de 70 razas es manejable, FAISS fue seleccionado como una decisión de diseño a prueba de futuro, ya que puede escalar fácilmente a millones de vectores sin una degradación significativa en el tiempo de inferencia.
- **Métrica de Distancia:** FAISS soporta métricas como la **Distancia Euclidiana** y la **Distancia Coseno**, esenciales para cuantificar la similitud entre los *embeddings*.

DESARROLLO

Etapa 1: Buscador de Imágenes por Similitud

El objetivo de esta etapa fue crear un sistema de **Recuperación de Imágenes Basada en Contenido (CBIR)** que, a partir de una imagen de consulta, identifique las 10 imágenes más similares dentro del *dataset* de 70 razas de perros.

Creación de la Base de Datos Vectorial

El primer paso crucial fue la conversión del *dataset* de imágenes a un formato que permitiera la comparación numérica eficiente: **vectores de características** (*embeddings*).

1. **Extracción de *Embeddings*:**
    - Se utilizó el modelo **ResNet50** pre-entrenado en **ImageNet** (a través de la librería **Tensor Flow**) como **extractor de características**.
    - Se eliminó la capa final de clasificación del ResNet50. La salida utilizada fue el vector de la capa de *Global Average Pooling*, que produce un **vector de *embedding* denso** (de 2048 dimensiones). Este vector captura las características semánticas y visuales de la imagen.
    - Cada imagen del *dataset* (previamente preprocesada con redimensionamiento y normalización) se pasó a través de este modelo para obtener su *embedding* correspondiente.
2. **Indexación Vectorial con FAISS:**
    - Los vectores de *embeddings* generados se indexaron utilizando **FAISS**. Se optó por un índice optimizado para manejar las consultas de vecinos más cercanos (*k*-NN*).
    - Se almacenó una **matriz de metadatos** paralela que mapea cada *embedding* en el índice de FAISS a su ruta de archivo original y a su etiqueta de raza, permitiendo la recuperación del *ground truth* y la imagen visual en la interfaz.

Desarrollo de la Aplicación en Gradio

Se desarrolló una interfaz de usuario interactiva utilizando la librería **Gradio**, diseñada para demostrar la funcionalidad del Buscador de Similitud:

1. **Interfaz:** La aplicación se configuró con un componente de entrada que permite al usuario subir una imagen de perro. El componente de salida principal consistió en un *display* para la imagen de entrada y una galería de imágenes para mostrar las 10 imágenes recuperadas.
2. **Flujo de Inferencia:** Cuando el usuario sube una imagen, la aplicación ejecuta los siguientes pasos en secuencia:
    - **Procesamiento:** La imagen de entrada es preprocesada (redimensionada y normalizada) para que coincida con el formato de entrenamiento del ResNet50.
    - **Extracción de *Embedding*:** Se pasa la imagen a través del modelo ResNet50 para obtener su vector de 2048 dimensiones.
    - **Consulta FAISS:** El *embedding* de consulta se utiliza en el índice de FAISS para encontrar los **10 índices más cercanos** utilizando la **Distancia Euclidiana** como métrica de similitud.
    - **Recuperación de Imágenes:** Usando los 10 índices devueltos por FAISS, se recuperan las rutas de las imágenes asociadas desde la matriz de metadatos y se muestran en la galería de Gradio.

Clasificación Basada en Similitud y Evaluación (NDCG@10)

Esta etapa completó la funcionalidad de clasificación y estableció la métrica de referencia para la calidad de la búsqueda.

1. **Clasificación por Voto Mayoritario:**
    - A partir de las 10 imágenes recuperadas de la base de datos vectorial, se extrajeron sus **etiquetas de raza correspondientes**.
    - Se implementó una lógica de **"voto mayoritario"** (majority vote): la raza con la mayor frecuencia entre los 10 resultados fue seleccionada como la **raza predicha** para la imagen de entrada.
    - Esta predicción se mostró en la interfaz de Gradio junto a las imágenes recuperadas.
2. **Métrica de Evaluación (NDCG@10):**
    - Para evaluar el rendimiento del sistema de recuperación de imágenes de manera objetiva, se preparó un **conjunto de prueba** consistente en 5 a 10 imágenes por raza, excluidas del conjunto de búsqueda.
    - Para cada imagen de prueba, se ejecutó la búsqueda y se calculó la métrica **Normalized Discounted Cumulative Gain (NDCG)** en k=10.
    - La métrica **NDCG@10** es esencial porque considera tanto la relevancia (si el resultado es la misma raza) como la posición del resultado. Los resultados correctos que aparecen más arriba en la lista reciben una puntuación más alta.

Etapa 2: Entrenamiento y Comparación de Modelos de Clasificación

El objetivo de esta etapa fue mejorar el rendimiento del *pipeline* de Visión por Computadora al entrenar modelos de clasificación específicos para el dominio canino, reemplazando el uso del extractor de características genérico (ResNet50 de ImageNet).

Se entrenaron dos modelos de clasificación distintos para evaluar cuál ofrecía una representación de características más discriminativa para las 70 razas de perros:

Modelo A: Transfer Learning (ResNet18 Fine-Tuned)

El modelo **ResNet18** (arquitectura de **Redes Residuales**) fue elegido por su equilibrio entre profundidad y eficiencia computacional. Se aplicó la técnica de **Transfer Learning** siguiendo el siguiente procedimiento:

1. **Inicialización:** Se cargó el ResNet18 con pesos pre-entrenados en ImageNet.
2. **Modificación de la Capa de Salida:** Se eliminó la capa de clasificación original (de 1000 clases) y se la reemplazó por una nueva capa densa con **70 salidas**, correspondiente al número de razas en el *dataset*.
3. ***Fine-Tuning* Estratégico:** Se implementó una estrategia de *fine-tuning* donde inicialmente se congelaron las capas convolucionales tempranas para proteger las características de bajo nivel. Luego, se **descongelaron todas las capas** y se entrenaron con una **tasa de aprendizaje muy baja** para ajustar ligeramente los pesos

pre-entrenados al nuevo dominio, asegurando que las nuevas características aprendidas fueran específicas de las razas de perros.

4. **Balanceo en Entrenamiento:** Durante el entrenamiento, se utilizó la estrategia de balanceo definida en la Metodología (Submuestreo y **Data Augmentation**) para las clases subrepresentadas.

Modelo B: CNN *Custom* (Entrenamiento desde Cero)

Para fines comparativos, se diseñó y entrenó un modelo **CNN *Custom*** desde cero. Este modelo generalmente consta de un conjunto de capas convolucionales y de *pooling* (ej., 3 a 5 bloques) seguidas de capas densas, inicializado con pesos aleatorios. Este modelo sirvió como **línea base** para cuantificar el beneficio del Transfer Learning (Modelo A) sobre el simple entrenamiento a partir de datos limitados.

- **Arquitectura:** La red se compone de **cinco bloques convolucionales** secuenciales, aumentando la profundidad de los filtros de **32** a **256**. Cada bloque convolucional utiliza **kernel_size=3** y **padding=1**, y está seguido por **Batch Normalization** y una función de activación ReLU, terminando con una capa **Max Pooling** para reducir la dimensión espacial. La imagen de entrada de 224x224 se reduce hasta una salida de **7x7** en el último bloque convolucional.
- **Capas Densas:** La salida se aplana a un vector de **12,544** dimensiones (256 filtros * 7x7) y se conecta a una capa densa oculta de **2048 neuronas**, seguida de la capa de clasificación final de **70 clases**.
- **Estrategias de Regularización:** Dada la profundidad de la red (que aumenta el riesgo de overfitting), se implementaron ajustes críticos de regularización:
    - **Dropout:** Se aplicó una capa de nn.Dropout(p=0.5) antes de la capa de salida.
    - **Weight Decay (Regularización L2):** Se implementó una regularización L2 con un valor de 0.0001 en el optimizador **Adam** para penalizar los pesos grandes y mejorar la capacidad de generalización del modelo.
- **Entrenamiento:** Este modelo fue entrenado durante 35 épocas, con una tasa de aprendizaje inicial de 0.0005, utilizando la función de pérdida **nn.CrossEntropyLoss** y aplicando la estrategia de balanceo de clases por **Data Augmentation** y **Undersampling** definida en la Metodología.

Integración y Selección en la Aplicación Gradio

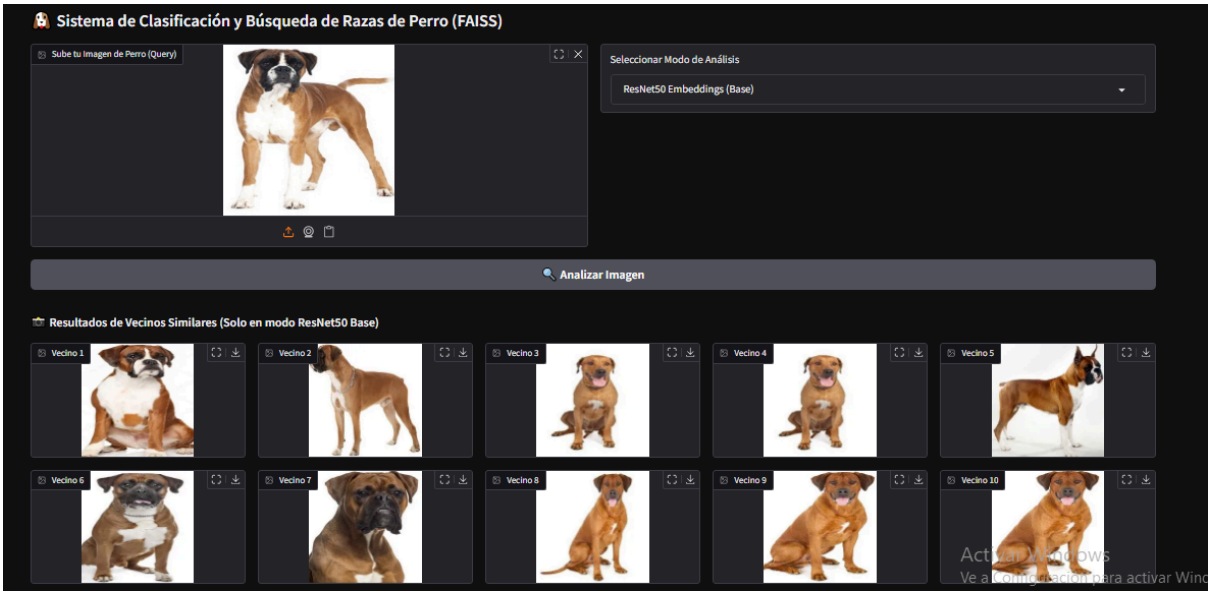
Una vez que el **ResNet18 *fine-tuned*** (Modelo A) y el **CNN *Custom*** (Modelo B) fueron entrenados, el sistema de búsqueda por similitud de la Etapa 1 fue actualizado para aprovechar las representaciones de características mejoradas.

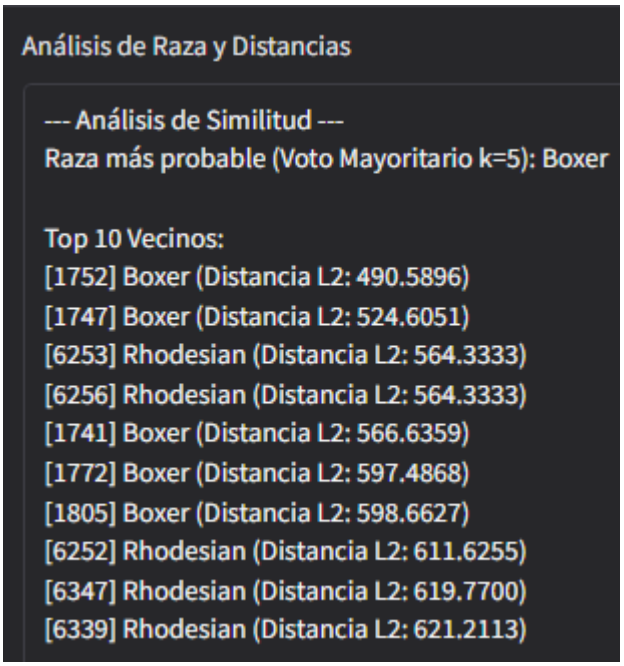
- **Actualización de Extracción de *Embeddings*:** Se integró la lógica en la función de *backend* de la aplicación Gradio para que pudiera generar *embeddings* utilizando el ResNet50 original, el Modelo A o el Modelo B.
- **Componente de Selección:** Se añadió un **menú desplegable (combo)** a la interfaz de Gradio. Este componente permite al usuario seleccionar dinámicamente el **Modelo de *Embedding*** deseado antes de realizar la búsqueda por similitud.

- **Flujo Actualizado:** La elección del usuario determina qué modelo (*ResNet50* original, *ResNet18 Fine-Tuned* o *CNN Custom*) se carga para procesar la imagen de consulta y generar el vector, el cual luego se pasa a FAISS para la búsqueda *k*-NN. Esta integración permitió una **comparación visual en tiempo real** de la calidad de los *embeddings* generados por cada modelo.

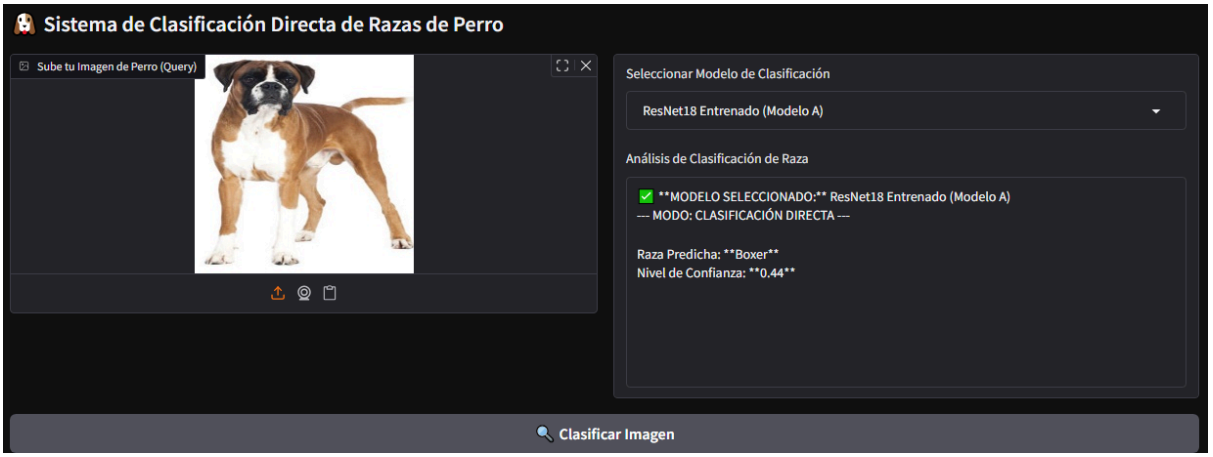
RESULTADOS

RESNET50

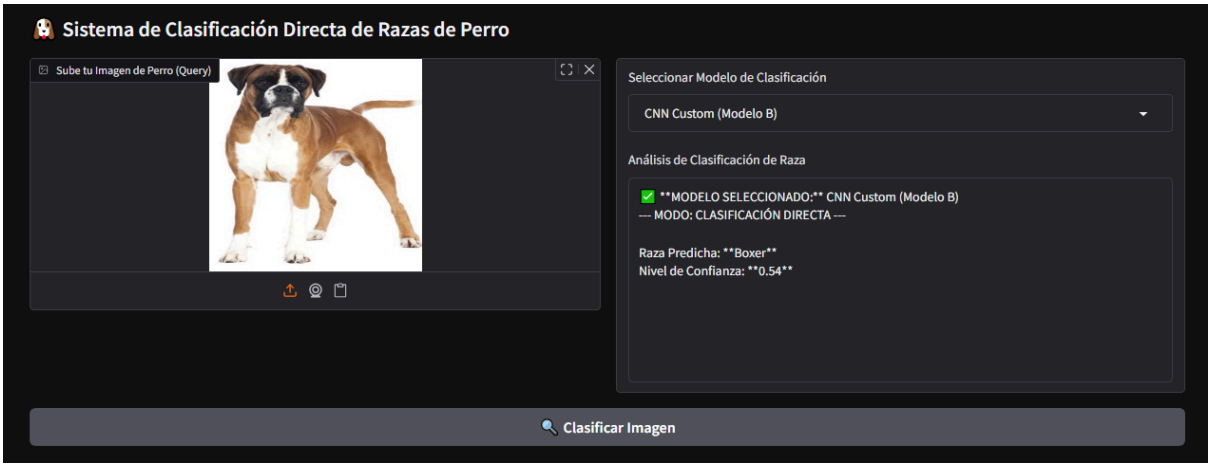




RESNET18


Sistema de Clasificación Directa de Razas de Perro
Sube tu Imagen de Perro (Query)
Seleccionar Modelo de Clasificación
ResNet18 Entrenado (Modelo A)
Análisis de Clasificación de Raza
MODELO SELECCIONADO: ResNet18 Entrenado (Modelo A)
--- MODO: CLASIFICACIÓN DIRECTA ---
Raza Predicha: **Boxer**
Nivel de Confianza: **0.44**
Clasificar Imagen

CUSTOM


Sistema de Clasificación Directa de Razas de Perro
Sube tu Imagen de Perro (Query)
Seleccionar Modelo de Clasificación
CNN Custom (Modelo B)
Análisis de Clasificación de Raza
MODELO SELECCIONADO: CNN Custom (Modelo B)
--- MODO: CLASIFICACIÓN DIRECTA ---
Raza Predicha: **Boxer**
Nivel de Confianza: **0.54**
Clasificar Imagen