UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

RECONOCIMIENTO Y CLASIFICACIÓN DE RAZAS DE PERROS

JAIME LÓPEZ DE LARRINZAR GÓMEZ

TUTOR: DAVID DOMÍNGUEZ CARRETA

NOVIEMBRE 2024

Resumen

Este Trabajo de Fin de Grado presenta el desarrollo de un algoritmo de clasificación de razas de perros a partir de imágenes utilizando técnicas de aprendizaje profundo en Python. El objetivo principal del proyecto es identificar con precisión la raza de un perro dado, utilizando un conjunto de datos de imágenes etiquetadas. Para ello, se han implementado y comparado diferentes arquitecturas de redes neuronales convolucionales preentrenadas, como MobileNetV2, ResNet50, ResNet101, ResNet152 y VGG16.

Además, se ha incorporado la técnica de Curriculum Learning, un enfoque de aprendizaje automático que consiste en presentar gradualmente al modelo subconjuntos de datos organizados en niveles de dificultad creciente, lo cual facilita una mejora progresiva en la precisión de la clasificación. Las imágenes de entrenamiento han sido preprocesadas para optimizar el rendimiento del modelo, y el proceso de entrenamiento se ha llevado a cabo ajustando diversos hiperparámetros, como la tasa de aprendizaje, el número de épocas, el porcentaje de dropout y el tamaño de batch. Los resultados obtenidos han sido evaluados mediante la métrica de precisión en un conjunto de validación, destacando las cinco mejores configuraciones.

El análisis muestra que la arquitectura MobileNetV2, con un aprendizaje ajustado y utilizando Curriculum Learning, ofrece el mejor rendimiento para este problema. MobileNetV2 obtiene un 90% de precisión, mientras que el segundo con mejor rendimiento, Resnet50, obtiene un 50%. No solo eso, sino que también, supera en tiempo y coste computacional a todos los modelos con una mejora de tiempo de hasta 6 veces respecto a otros de los modelos con menor tiempo de entrenamiento, ResNet101.

Finalmente, se discuten las posibles mejoras y trabajos futuros, como la implementación de técnicas de aumento de datos.

Palabras clave

Aprendizaje profundo, Clasificación de imágenes, Redes neuronales convolucionales, Reconocimiento de razas de perros, MobileNetV2, ResNet, VGG16, Procesamiento de imágenes, TensorFlow, Python, Keras, Preprocesamiento de imágenes, Arquitecturas de redes profundas, Optimización de hiperparámetros, learning rate, batch size, dropout, épocas, Curriculum Learning.

Summary

This bachelor's Thesis focuses on developing a deep learning-based algorithm in Python for classifying dog breeds from images. The project's main objective is to accurately determine the breed of a dog using a dataset of labeled images. To accomplish this, several pre-trained convolutional neural network architectures, including MobileNetV2, ResNet50, ResNet101, ResNet152, and VGG16, were implemented and compared.

The research also integrates the Curriculum Learning technique, a machine learning approach that progressively introduces data subsets arranged by increasing difficulty levels. This strategy helps improve classification accuracy over time. The training images underwent preprocessing to enhance model performance, and the training process involved fine-tuning hyperparameters such as learning rate, epochs, dropout rate, and batch size. Accuracy metrics were used to evaluate the results on a validation set, identifying the top five model configurations.

Findings reveal that MobileNetV2, combined with optimized learning parameters and Curriculum Learning, achieves the best performance. It delivers up to 40% greater accuracy compared to ResNet50, the second-best model. Additionally, MobileNetV2 excels in efficiency, reducing training by 6 compared to ResNet101, one of the fastest tested models.

The study concludes by exploring potential enhancements, such as employing data augmentation techniques, for further improvement and future research directions.

Keywords

Deep learning, Image classification, Convolutional neural networks, Dog breed recognition, MobileNetV2, ResNet, VGG16, Image processing, TensorFlow, Python, Keras, Image preprocessing, Deep network architectures, Hyperparameter optimization, learning rate, batch size, dropout, epochs, Curriculum Learning.

ÍNDICE

Índice de tablas

Índice de Figuras

1. Introducción

1.1. Contexto y Justificación

El reconocimiento de imágenes mediante técnicas de inteligencia artificial ha experimentado un crecimiento exponencial en los últimos años, impulsado por avances en el aprendizaje profundo y la disponibilidad de grandes conjuntos de datos. En este contexto, la clasificación de razas de perros a partir de imágenes se ha convertido en un desafío interesante y relevante, tanto para la industria como para la investigación académica. La capacidad de identificar automáticamente la raza de un perro tiene aplicaciones prácticas en diversas áreas, como en clínicas veterinarias, el desarrollo de aplicaciones móviles para propietarios de mascotas, e incluso en el ámbito de la seguridad.

A pesar de los avances, la clasificación precisa de razas de perros presenta varios desafíos técnicos. La variabilidad en la apariencia de los perros, incluso dentro de la misma raza, y la similitud visual entre diferentes razas complican la tarea para los modelos de visión artificial. Para abordar estas complejidades, este proyecto ha incorporado la técnica de Curriculum learning, un enfoque de aprendizaje progresivo en el que el modelo entrena con conjuntos de datos de dificultad creciente. Este enfoque se ha aplicado con MobileNetV2 en 40 razas de perros, realizando pruebas en diferentes configuraciones: (1) una sola fase de entrenamiento con las 40 razas, una configuración de cuatro fases que añade progresivamente 10 razas en cada fase, y (3) un esquema de 10 fases con 4 razas añadidas en cada fase. Las razas de cada fase fueron cuidadosamente seleccionadas para comenzar con subgrupos de razas fáciles de diferenciar, facilitando así un aprendizaje más efectivo y mejorando la capacidad de generalización del modelo.

Además de la complejidad intrínseca de la clasificación multiclase, la necesidad de grandes volúmenes de datos etiquetados [1] y la elección adecuada de arquitecturas de redes neuronales, como MobileNetV2, son factores vitales para el éxito de estos sistemas.

1.2. Motivación

Mi interés por la inteligencia artificial, y más específicamente por la visión artificial, me llevó a elegir este tema para mi Trabajo de Fin de Grado. A pesar de no tener conocimientos previos sobre redes neuronales, vi en este proyecto una oportunidad perfecta para aprender y adentrarme en un campo que me apasiona. La capacidad de las máquinas para "ver" y comprender imágenes es un área que encuentro fascinante y con mucho potencial para transformar la sociedad y la forma en que interactuamos con la tecnología.

Identificar la raza de un perro tiene un enorme potencial en muchos ámbitos de nuestra vida diaria. Desde aplicaciones móviles que ayudan a los amantes de los animales, hasta herramientas para veterinarios y el apoyo a refugios de rescate, esta tecnología puede marcar una gran diferencia. Sin embargo, para que funcione en dispositivos como teléfonos inteligentes, se necesitan sistemas muy precisos y eficientes que puedan aprovechar al máximo los recursos disponibles. Por eso, investigar y aplicar soluciones como MobileNetV2 y técnicas de optimización en tiempo real es clave para superar estos retos.

Además, este tipo de investigación responde a una necesidad cada vez más importante: hacer que la tecnología sea accesible para todos. Poder identificar la raza de un perro a través de una simple fotografía no solo es práctico, sino también educativo, ayudando a las personas a valorar la increíble diversidad que existe entre las razas caninas. Combinando los avances de la inteligencia artificial con aplicaciones útiles, este trabajo tiene el potencial de mejorar la vida diaria de muchas personas, fomentando el conocimiento y el cuidado responsable de nuestras queridas mascotas.

Una motivación adicional para este trabajo proviene de una experiencia personal: tengo un perro cuya raza desconocía, lo que me llevó a pensar en cómo la tecnología podría ayudar a resolver este tipo de incógnitas. Este proyecto, por tanto, no solo representa un reto académico, sino también una solución potencial a un problema cotidiano, que al igual que yo, puedan tener otras personas.

Finalmente, este trabajo también refuerza y habilita en mi deseo de seguir formándome y especializándome en el campo de la inteligencia artificial. Estoy convencido de que este proyecto me proporcionará las bases necesarias para futuras investigaciones y desarrollos en este campo el cual encuentro tan interesante como útil.

1.3. Objetivos

Objetivo general:
Desarrollar un modelo de aprendizaje profundo capaz de clasificar con precisión diversas razas de perros a partir de imágenes, utilizando diferentes arquitecturas de redes neuronales y aplicando técnicas avanzadas de aprendizaje como el Curriculum learning.

Objetivos específicos:

1. Preprocesar y preparar un conjunto de datos de imágenes etiquetadas para entrenar y evaluar los modelos de clasificación, asegurando la normalización y consistencia de las imágenes en cada fase de dificultad.

2. Explorar y comparar diferentes arquitecturas de redes neuronales convolucionales preentrenadas, como MobileNetV2, ResNet y VGG16, para seleccionar la más adecuada para la tarea de clasificación de razas de perros.

3. Implementar la técnica de Curriculum learning, estructurando el entrenamiento en fases de dificultad progresiva para mejorar la precisión y generalización del modelo, evaluando el impacto de entrenar en varias etapas frente a un entrenamiento único.

4. Optimizar los hiperparámetros de los modelos para maximizar la precisión de clasificación, ajustando variables como la tasa de aprendizaje, el porcentaje de dropout, el número de épocas y el tamaño de batch.

5. Evaluar y comparar el rendimiento de los modelos seleccionados utilizando métricas de precisión y pérdida en un conjunto de validación, analizando el efecto de las fases de dificultad en el rendimiento del modelo.

6. Proponer mejoras y trabajos futuros basados en los resultados obtenidos, considerando la implementación de técnicas adicionales como el aumento de datos y la optimización del modelo para aplicaciones móviles.

2. Estado del arte

En los últimos años, la inteligencia artificial, y en especial el aprendizaje profundo, ha revolucionado la visión artificial, abriendo nuevas oportunidades en tareas complejas como la clasificación de imágenes, la detección de objetos y la segmentación semántica. Dentro de este campo, las redes neuronales convolucionales (CNNs) se han convertido en una herramienta esencial, ya que tienen la capacidad de identificar y aprender patrones visuales directamente de los datos sin procesar, como las imágenes [2][3].

A diferencia de los métodos anteriores, que dependían de características diseñadas manualmente, las CNNs permiten que el modelo aprenda y construya sus propias jerarquías de características visuales. Esto ha hecho que sean especialmente eficaces en tareas como la clasificación de razas de perros, donde es fundamental captar detalles de textura, color y forma que pueden ser muy sutiles, incluso para el ojo humano [4][5].

La clasificación de razas caninas tiene un amplio potencial en distintos sectores, incluyendo aplicaciones móviles, herramientas de apoyo para clínicas veterinarias y el trabajo de organizaciones de rescate animal. Este tipo de aplicaciones requiere sistemas precisos y eficientes que puedan funcionar en dispositivos de recursos limitados, como los teléfonos móviles. Afrontar este desafío abre la puerta a explorar arquitecturas optimizadas, como MobileNetV2, y métodos avanzados de optimización que resultan particularmente efectivos en contextos que exigen un rendimiento en tiempo real.

Finalmente, esta investigación también responde a la creciente necesidad de accesibilidad tecnológica. La posibilidad de identificar la raza de un perro a partir de una fotografía sencilla proporciona al usuario una herramienta práctica que, además, fomenta el interés y la educación sobre la diversidad de razas caninas. Esta perspectiva accesible y socialmente orientada de la inteligencia artificial permite que la investigación en redes neuronales y visión por computadora no se limite a los avances técnicos, sino que también tenga un impacto directo en la vida diaria de las personas, ayudando a identificar razas y a mejorar el entendimiento general de las características y necesidades de cuidado de diferentes razas de perros.

Así, explorar la clasificación de razas de perros no solo contribuye al avance en conocimientos técnicos, sino que también impulsa el desarrollo de aplicaciones prácticas y accesibles en la vida cotidiana, acercando la inteligencia artificial a una audiencia más amplia y diversa.

2.1 Redes Neuronales Convolucionales (CNNs) en la Clasificación de Imágenes

Las redes neuronales convolucionales (CNNs) surgieron como una innovación en la década de 1980, pero su verdadero auge llegó en 2012 con el modelo AlexNet, que causó gran impacto en el concurso ImageNet gracias a una precisión sin precedentes en la clasificación de imágenes. Desde entonces, las CNNs han evolucionado de manera exponencial, y se han desarrollado arquitecturas que no solo incrementan la precisión, sino que también optimizan la eficiencia y adaptabilidad de los modelos en distintos contextos. A continuación, exploramos algunas de las más influyentes:

MobileNetV2: Esta arquitectura fue desarrollada en 2018 por Sandler et al. para satisfacer la necesidad de modelos eficientes en términos de recursos. MobileNetV2 está diseñado para optimizar la precisión manteniendo un bajo número de parámetros, lo cual es ideal para aplicaciones en dispositivos móviles y entornos con recursos limitados. Su enfoque innovador incluye el uso de convoluciones de profundidad separable junto con conexiones residuales, lo que permite un equilibrio notable entre eficiencia y rendimiento.

ResNet (Red Residual): La introducción de ResNet en 2015 por He et al. marcó un hito en la arquitectura de redes neuronales profundas al presentar los "bloques residuales". Esta innovación permite entrenar redes mucho más profundas (como la ResNet152, que alcanza hasta 152 capas) sin sufrir los típicos problemas de degradación de precisión que suelen acompañar a modelos tan grandes. Los bloques residuales hacen posible que el modelo mantenga la precisión al aprender de forma eficiente incluso con redes muy extensas, convirtiendo a ResNet en una referencia clave para las tareas avanzadas de visión artificial.

VGG16: Esta arquitectura fue diseñada por Simonyan y Zisserman [6] en 2014 y se destaca por su estructura clara y eficaz. Las redes VGG emplean una serie de capas convolucionales seguidas de capas completamente conectadas, lo que permite captar patrones visuales complejos y, al mismo tiempo, generalizar en diversas tareas de clasificación. En particular, VGG16 ha ganado popularidad debido a su balance entre capacidad de generalización y precisión, lo que la convierte en una opción confiable para distintas aplicaciones de clasificación de imágenes.

Curriculum Learning: Una de las técnicas avanzadas que se ha integrado es el Curriculum Learning [7], una estrategia de aprendizaje progresivo. En este caso, el modelo fue entrenado en tres configuraciones de dificultad creciente para clasificar 40 razas de perros. La primera configuración involucró una fase única con todas las razas, logrando un 82.36% de precisión. La segunda usó 4 fases, añadiendo 10 razas progresivamente, lo cual elevó la precisión a un 83.97%. La tercera configuración, con 10 fases de 4 razas cada una, alcanzó el máximo rendimiento, con un 87.00% de precisión. Este enfoque, que imita el proceso de aprendizaje humano, permite que el modelo adquiera conocimientos básicos sobre las razas más fácilmente distinguibles, antes de enfrentarse a aquellas con mayores similitudes visuales.

2.2 Clasificación de Razas de Perros

La clasificación de razas de perros dentro del campo de la visión artificial plantea retos únicos. La gran variedad de características físicas y las similitudes visuales entre algunas razas hacen de esta una tarea compleja. Sin embargo, las redes neuronales convolucionales (CNNs) han abierto la puerta a soluciones cada vez más precisas y robustas, y algunos estudios han resultado particularmente relevantes:

Stanford Dog Dataset: Este conjunto de datos es uno de los más utilizados y contiene imágenes de 120 razas de perros. Ha servido como base para evaluar y entrenar modelos de CNN, demostrando que estas redes pueden diferenciar entre razas con alta precisión, incluso entre aquellas que parecen similares. Además, permite implementar técnicas como el Curriculum learning, una estrategia que guía al modelo a través de etapas de dificultad progresiva para que aprenda de manera más eficiente.

Transfer Learning para la Clasificación de Razas: Entrenar un modelo desde cero suele requerir grandes cantidades de datos y mucha capacidad computacional. Por eso, se recurre a menudo al transfer learning, que consiste en adaptar modelos previamente entrenados en conjuntos de datos amplios y generales, como ImageNet, para que trabajen en una tarea específica, como identificar razas de perros. Este enfoque permite alcanzar niveles altos de precisión y sacar el máximo partido de los recursos disponibles, ya que el modelo aprovecha conocimientos previos y los ajusta para diferenciar entre razas.

Aplicaciones Comerciales y Herramientas para Usuarios: En los últimos años, varias aplicaciones móviles y herramientas en línea han comenzado a ofrecer la posibilidad de identificar la raza de un perro a partir de una foto. Estas aplicaciones suelen usar modelos de CNN avanzados y técnicas adicionales, como la aumentación de datos y el aprendizaje por fases.

2.3 Literatura reciente

Raduly et al. (2018) [8] investigaron el uso de redes neuronales convolucionales (CNN) para clasificar razas de perros, priorizando el desarrollo de un modelo capaz de identificar razas con precisión en imágenes con detalles visuales complejos. Durante sus experimentos, alcanzaron una precisión de validación del 85% empleando técnicas de ajuste fino en modelos preentrenados y un meticuloso preprocesamiento de las imágenes. Este enfoque logró superar significativamente los métodos tradicionales, estableciendo una base robusta para aplicaciones prácticas en la clasificación de razas.

Valarmathi et al. (2023) [9] llevaron a cabo un análisis comparativo empleando arquitecturas híbridas que combinan modelos CNN como EfficientNet y Xception, con el propósito de mejorar la precisión en la clasificación de razas. Esta estrategia alcanzó un 92% de precisión en pruebas

realizadas con datasets que incluían más de 100 razas. La integración de características extraídas de diferentes arquitecturas demostró su efectividad para superar los métodos tradicionales, permitiendo capturar rasgos visuales complejos y optimizar la clasificación.

Gunaranjan et al. (2024) [10] exploraron el uso de técnicas de aprendizaje por transferencia aplicadas a modelos CNN, específicamente con Inception V3, para la clasificación de razas de perros. Al ajustar el modelo utilizando un conjunto de datos preentrenado en ImageNet, lograron una precisión del 88% al clasificar un dataset con 120 razas. Este enfoque destacó por reducir el tiempo de entrenamiento en un 40% en comparación con un modelo desarrollado desde cero, demostrando el valor del aprendizaje por transferencia para optimizar recursos y mantener una alta precisión con volúmenes de datos moderados.

Saglietti et al. (2022) [11] estudiaron cómo utilizando Curriculum Learning, es decir; comenzar con ejemplos más sencillos puede disminuir el tiempo de entrenamiento hasta en un 20% en tareas con datos complejos. Además, la implementación de regularización entre etapas permitió incrementar la precisión en las pruebas hasta un 5% en comparación con métodos convencionales. Esto evidencia que el aprendizaje curricular no sólo acelera el entrenamiento, sino que también potencia la capacidad de generalización de las redes neuronales.

2.4 Desafíos y Oportunidades

Aunque la clasificación de razas de perros ha progresado notablemente, todavía existen desafíos clave que limitan el rendimiento de los modelos y plantean áreas de oportunidad para futuras investigaciones y mejoras:

Similitud Visual entre Razas: Muchas razas de perros comparten características visuales sutiles, lo que puede confundir incluso a modelos avanzados. Para abordar esto, algunos enfoques incluyen el uso de arquitecturas más sofisticadas o técnicas especializadas, como redes de atención que resaltan diferencias clave en las imágenes, o redes neuronales siamesas que están diseñadas para identificar y aprender variaciones mínimas entre clases similares.
Diversidad en las Condiciones de las Imágenes: Los modelos suelen entrenarse con imágenes de alta calidad y en entornos controlados, pero en aplicaciones prácticas, las imágenes varían ampliamente en términos de iluminación, ángulo y fondo. Estas variaciones pueden reducir la precisión en entornos reales, resaltando la importancia de integrar técnicas como la aumentación de datos, que crea un entrenamiento más robusto al simular condiciones variables.
Desbalance de Clases: Los conjuntos de datos suelen estar desequilibrados, con algunas razas representadas por muchas más imágenes que otras. Esto puede inducir un sesgo en el modelo hacia las razas con más datos, afectando la capacidad de clasificar correctamente las razas menos comunes. Estrategias como el reequilibrio de clases y el ajuste de pérdidas por clase son comunes para mitigar este factor.

3. Diseño

3.1 Requisitos del Sistema

El diseño del sistema de clasificación de razas de perros se fundamenta en una serie de requisitos funcionales y no funcionales para guiar su desarrollo, incluyendo el uso de Curriculum Learning para optimizar la precisión en clasificaciones complejas.

3.1.1 Requisitos Funcionales

Entrada y preprocesamiento de imágenes: El sistema debe aceptar imágenes de distintos tamaños y formatos, y realizar preprocesamiento (redimensionamiento, normalización, etc.) para garantizar una entrada uniforme que permita un aprendizaje óptimo en cada fase.

Clasificación de imágenes: El sistema debe clasificar con precisión una imagen de un perro dentro de una de las razas predefinidas. Para esto, se aplica un enfoque de aprendizaje en fases, donde el modelo se entrena inicialmente con razas de fácil diferenciación y gradualmente se exponen nuevas razas.

Entrenamiento en fases progresivas: Se facilita el entrenamiento del modelo mediante Curriculum Learning, utilizando arquitecturas preentrenadas como MobileNetV2, ResNet y VGG16. Esta técnica ayuda a adaptar la capacidad del modelo de forma progresiva, añadiendo nuevas razas en cada fase.

Evaluación y validación acumulativa: El sistema debe incluir un proceso de validación que permita evaluar la precisión del modelo en cada fase del aprendizaje. La validación acumulativa proporciona un análisis continuo de rendimiento y ajuste.

Predicción en tiempo real: El sistema debe ser capaz de realizar predicciones en tiempo real sobre imágenes nuevas no vistas durante el entrenamiento, aprovechando el aprendizaje incremental para lograr una precisión sólida en la predicción de diversas razas.

3.1.2 Requisitos No Funcionales

Eficiencia: El sistema debe estar optimizado para tiempos de respuesta adecuados, tanto en la fase de entrenamiento como en la inferencia. Curriculum Learning contribuye a la eficiencia, permitiendo que el modelo aprenda de manera gradual y optimice el uso de recursos en cada fase.

Escalabilidad: La arquitectura debe ser escalable para admitir la inclusión de nuevas razas o conjuntos de datos ampliados sin necesidad de rediseñar el sistema. Al estructurar el entrenamiento en fases, Curriculum Learning permite integrar nuevas clases progresivamente sin comprometer el rendimiento general.

3.2 Arquitectura del Sistema

El sistema de clasificación se ha diseñado siguiendo una arquitectura modular que facilita la extensión y el mantenimiento. Los componentes principales son:

Módulo de Preprocesamiento de Imágenes: Responsable de preparar las imágenes de entrada para su uso en el modelo. Incluye operaciones como redimensionar las imágenes a un tamaño fijo (224x224 píxeles en este caso) [12], normalizar los valores de los píxeles y convertir las imágenes a matrices NumPy. Estos pasos aseguran una entrada consistente y optimizada para el modelo.

Módulo de Modelos de Clasificación: Contiene las diferentes arquitecturas de redes neuronales convolucionales (CNNs) implementadas en el sistema. Las arquitecturas preentrenadas seleccionadas incluyen MobileNetV2, ResNet50, ResNet101, ResNet152 y VGG16, que sirven como base para el aprendizaje de características complejas necesarias para la clasificación de razas.

Módulo de Entrenamiento y Validación: Administra el proceso de entrenamiento y validación. Para mejorar la precisión y facilitar la generalización, este módulo integra Curriculum Learning, una técnica en la que las razas se introducen en fases progresivas de dificultad. Esta estrategia ayuda a que el modelo aprenda gradualmente, comenzando con subgrupos fáciles de diferenciar y añadiendo complejidad en cada fase.

Módulo de Predicción: Tras el entrenamiento, este módulo realiza predicciones sobre nuevas imágenes y proporciona una medida de la confianza en la clasificación. Es fundamental para aplicaciones en tiempo real y asegura que el sistema responda eficientemente a entradas externas.

3.3 Diagrama de Arquitectura

El siguiente diagrama ilustra la arquitectura general del sistema:

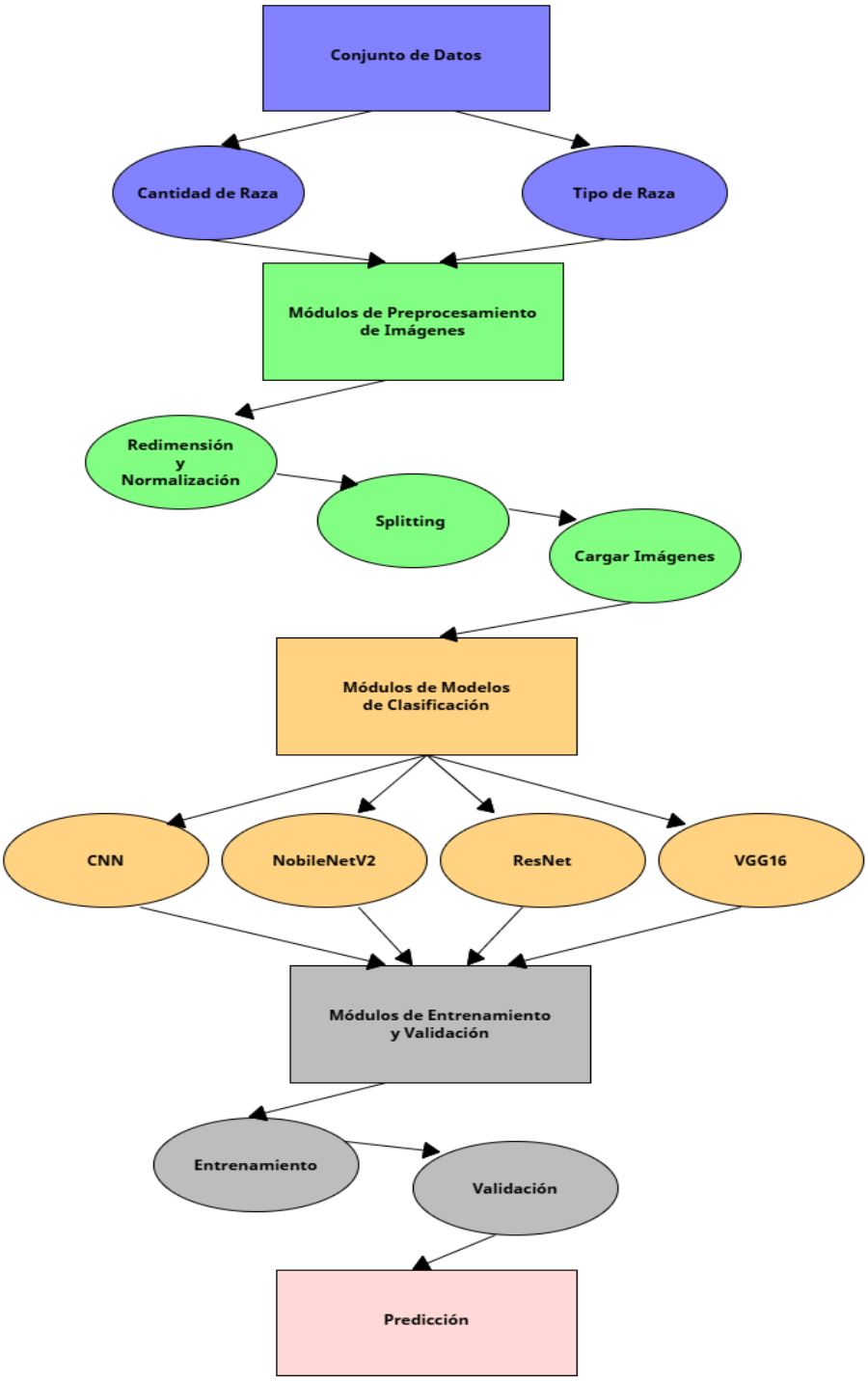

Figura 1: Diagrama de arquitectura general del sistema.

3.4 Flujo de Trabajo

El flujo de trabajo del sistema sigue los siguientes pasos:

Preprocesamiento de Datos: Las imágenes del conjunto de datos se cargan y se preprocesan para ajustarse al tamaño y formato requerido por las CNNs, asegurando uniformidad en las entradas.

Entrenamiento del Modelo: El modelo se entrena utilizando las imágenes preprocesadas, y se ajustan los hiperparámetros como la tasa de aprendizaje, el tamaño del lote y el número de épocas para optimizar el rendimiento. Se emplea la técnica de Curriculum Learning, donde las razas se introducen en fases progresivas, comenzando con las más fáciles de diferenciar. Esto permite que el modelo aprenda de manera gradual, mejorando su capacidad de generalización.

Evaluación: El modelo se evalúa en un conjunto de datos de validación para medir su precisión y otros indicadores de rendimiento. La validación acumulativa permite monitorear el progreso a medida que se añaden nuevas razas en cada fase, evaluando de manera continua la capacidad del modelo para adaptarse a conjuntos de datos más complejos.

Predicción: Una vez entrenado, el modelo se utiliza para predecir la raza de nuevas imágenes. Los resultados se presentan con una indicación de la confianza de la predicción, lo cual es fundamental para el usuario en aplicaciones reales.

Ajustes y Mejoras: En función de los resultados obtenidos, se pueden realizar ajustes en la arquitectura del modelo o en los datos para mejorar la precisión. Estos ajustes pueden incluir modificaciones en el Curriculum Learning o el uso de técnicas adicionales como el aumento de datos.

3.5 Herramientas y Tecnologías Utilizadas

Hardware:

Procesador (CPU): Intel Core i7-7500U a 2.70 GHz, 2 núcleos y 4 hilos.
Tarjeta Gráfica (GPU): NVIDIA GeForce 940MX con 2 GB de memoria GDDR5.
Memoria RAM: 12 GB DDR4.
Almacenamiento: HDD de 1 TB.
Sistema Operativo: Windows 10 Home, versión 21H1.

El hardware utilizado, aunque no es de gama alta, es adecuado para el desarrollo y entrenamiento de modelos de aprendizaje profundo de tamaño moderado. Para modelos más complejos o conjuntos de datos más grandes, podría ser necesario utilizar recursos adicionales, como servicios en la nube con GPU dedicadas.

Software:

Para la implementación de este sistema, se han utilizado las siguientes herramientas y tecnologías:

Anaconda y Colab: Utilizado para la gestión del entorno de desarrollo y la instalación de dependencias.
Pandas: Biblioteca utilizada para la manipulación de datos y la gestión de conjuntos de datos en formato CSV.
NumPy: Biblioteca fundamental para operaciones numéricas y manipulación de arrays.
Matplotlib: Biblioteca utilizada para la visualización de datos y la creación de gráficos que muestran los resultados del entrenamiento.
Scikit-learn: Biblioteca utilizada para la evaluación del rendimiento del modelo y el cálculo de métricas adicionales.
Python: Lenguaje de programación principal utilizado para el desarrollo de todo el sistema.
TensorFlow y Keras: Librerías para la implementación y entrenamiento de los modelos de aprendizaje profundo.

4. Implementación/Desarrollo

4.1 Preprocesamiento de Datos

El preprocesamiento de los datos es un paso crucial para garantizar que las imágenes estén en un formato adecuado para ser introducidas en el modelo de red neuronal. A continuación, se describen los pasos llevados a cabo en este proceso.

4.1.1. Carga de Datos

Las etiquetas de las imágenes fueron cargadas desde un archivo CSV utilizando la biblioteca de pandas. Este archivo contiene los identificadores de las imágenes y la raza correspondiente.

Se añadieron las extensiones .JPG a los identificadores de las imágenes para que coincidieran con los nombres de archivo de las imágenes en los directorios de entrenamiento y prueba [Apéndice A].

4.1.2 Selección de Razas

Para experimentar con diferentes niveles de complejidad en el modelo, se seleccionaron listas de 10 y 40 razas de perros. Esto permitió observar cómo el modelo se desempeñaba con un número creciente de clases.
Los resultados discutidos en este trabajo corresponden a la selección de 10 razas. Los datos obtenidos para 40 razas son de menor precisión de manera proporcional, obteniendo entre un 15-17% inferior para 40 razas. Con inferior se entiende la precisión máxima obtenida para 10 razas, es decir; 97%.

Solo se mantuvieron las imágenes correspondientes a las razas seleccionadas, eliminando las demás.

Por otro lado, también se realizaron 4 subconjuntos de 10 razas, los cuales representaban cada uno un tamaño de raza, es decir; el primero subconjunto contenía razas pequeñas tales como chihuahua o pug, un segundo subconjunto con tamaños de razas intermedias-pequeñas tales como basenji o Beagle, un tercer subconjunto con tamaños de razas intermedias-grandes tales como Golden retriever y, por último, un subconjunto con razas grandes tales como Saint Bernard o Great Dane. Las precisiones conseguidas respectivamente son: 86%, 85.5%, 88,5% y 93%. Estos valores son el resultado de la media de 10 entrenamientos por cada subconjunto

4.1.3 Distribución de Imágenes

Se realizó un análisis de los datos para contar y graficar la distribución de imágenes por cada raza seleccionada. Esto permitió identificar posibles desequilibrios en el conjunto de datos que podrían afectar el rendimiento del modelo tal y como se mencionó anteriormente.

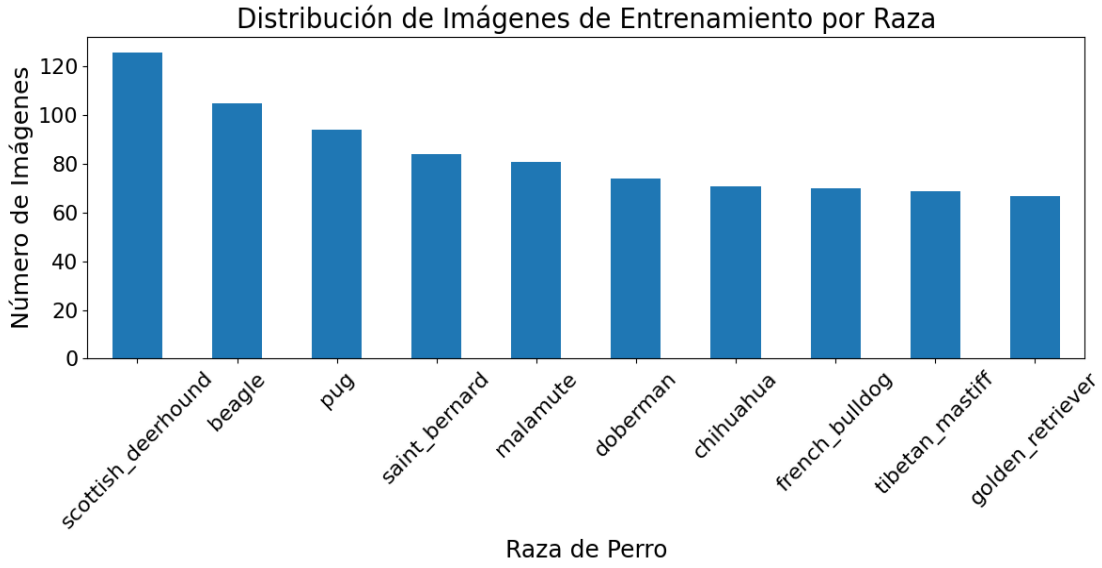

Figura 2: Distribución de las distintas razas de perro para el caso de 10 razas. [Apéndice B].

4.1.4 Preprocesamiento de Imágenes

Se definió una función para cargar y preprocesar las imágenes. Las imágenes fueron redimensionadas a 224x224 píxeles, que es el tamaño de entrada esperado por las arquitecturas de redes neuronales convolucionales utilizadas.

Además, se normalizaron los valores de los píxeles para que estuvieran en el rango [0, 1], lo cual es importante para la estabilidad del entrenamiento del modelo. [Apéndice C]

Con estabilidad del entrenamiento me refiero a que la normalización de los valores de los píxeles al rango [0,1] mejora la estabilidad y eficiencia del entrenamiento del modelo, evitando gradientes grandes y facilitando que el modelo aprenda patrones de mejor manera. Además, mantiene la compatibilidad con redes preentrenadas, haciendo más fácil su adaptación.

4.1.5 División del Conjunto de Datos

Se dividieron las imágenes en conjuntos de entrenamiento y validación, utilizando el 20% de las imágenes para validación. Esto permitió evaluar el rendimiento del modelo en datos que no fueron vistos durante el entrenamiento. [Apéndice D].

4.1.6 Codificación de Etiquetas

Las etiquetas de las razas fueron codificadas en un formato categórico utilizando la función to_categorical de Keras, lo que permitió que el modelo output un vector de probabilidades para cada clase. [Apéndice E].

4.2 Implementación de Modelos

4.2.1 Arquitecturas exploradas

Se utilizaron distintos modelos de arquitectura de red neuronal tales como una propia, VGG16, ResNet50, ResNet101, ResNet152 y MobileNetV2. En cada modelo, ajusté varios hiperparámetros para mejorar el rendimiento y obtener los mejores resultados posibles. Para el caso de MobileNetV2, se hicieron pruebas con la técnica Curriculum learning.

CNN Simple:

- **Tasa de aprendizaje:** Se probaron valores de 0.0005, 0.0002, 0.5 y 0.2. La tasa de 0.0002 fue la que ofreció el mejor resultado, alcanzando una precisión de validación de 27.14%.
- **Dropout:** Se evaluaron configuraciones con 0% y 40%. El dropout del 40% permitió una ligera mejora en la capacidad de generalización del modelo, aunque con limitaciones.
- **Tamaño de batch:** Se exploraron tamaños de batch de 16 y 64. El batch size de 64 proporcionó un equilibrio razonable entre estabilidad del entrenamiento y tiempo de ejecución, logrando los mejores resultados con esta configuración.
- **Número de épocas:** El modelo fue entrenado durante 8 épocas en cada configuración, lo que resultó suficiente para observar el comportamiento del aprendizaje sin incurrir en sobreentrenamiento.

Aunque la red convolucional propia no logró alcanzar niveles de precisión similares a los de modelos preentrenados como MobileNetV2, los resultados indican que el diseño base podría servir como punto de partida para experimentos adicionales, especialmente con mejoras en la arquitectura o con técnicas avanzadas como transfer learning o Curriculum learning.

MobileNetV2

- **Tasa de aprendizaje:** Una tasa de 0.0005 fue la más efectiva, alcanzando una precisión de validación del 97.02% [Tabla 1].
- **Dropout:** Se probó dropout del 20% y 40%, siendo 20% el que mejoró la capacidad de generalización.
- **Tamaño de batch:** Los tamaños variaron entre 16 y 64, con un batch de 32 logrando un equilibrio efectivo.
- **Número de épocas:** El modelo fue entrenado en 15 épocas para permitir un aprendizaje profundo sin sobreentrenamiento.

Dado que MobileNetV2 es el modelo con mejor rendimiento y precisión, se hicieron más pruebas con 40 razas. Las características del modelo eran las mismas exceptuando el batch size, que en este caso era de 128 dado que la muestra tendría mayor número de imágenes.

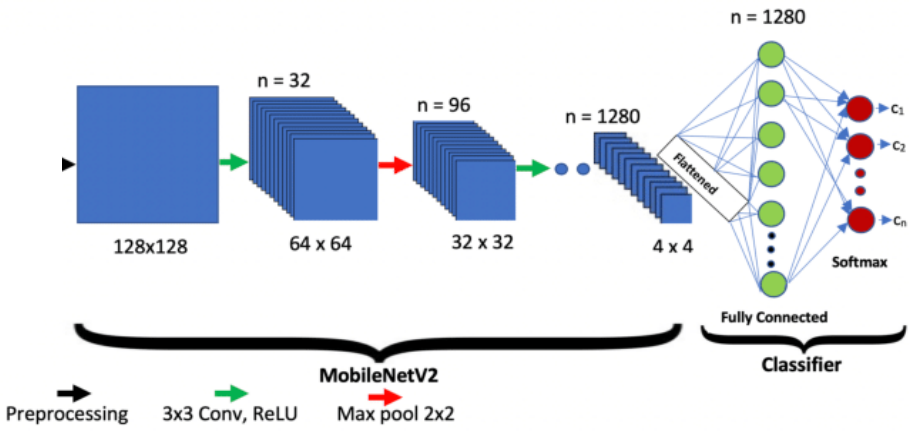

Figura 3: Representación visual de la arquitectura de la red neuronal MobileNetV2. [13]

ResNet50

- **Tasa de aprendizaje:** Ajusté la tasa de aprendizaje a valores entre 0.0001 y 0.5. La tasa de 0.0005 proporcionó resultados estables y una precisión de validación del 57.73% [Tabla 2].
- **Dropout:** Usé dropout del 0% al 40%, logrando mejores resultados con un dropout del 0%, que ayudó a preservar la información en las capas profundas.
- **Tamaño de batch:** El tamaño de batch de 16 ofreció el mejor rendimiento, aunque se probó con hasta 128.
- **Número de épocas:** Configuré 15 épocas para evaluar cómo el modelo se ajustaba progresivamente.

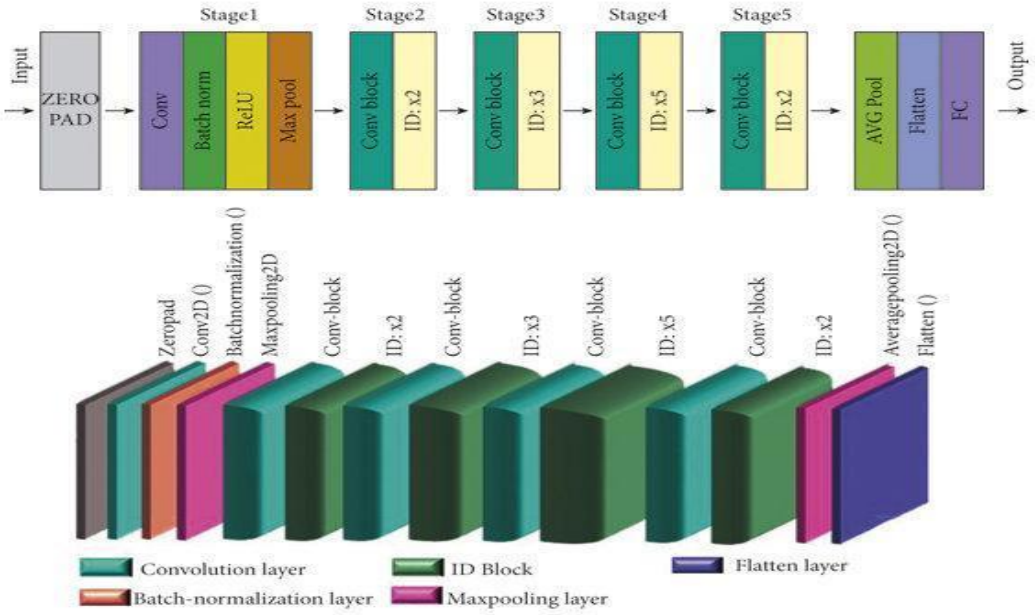

Figura 4: Representación visual de la arquitectura de la red neuronal ResNet 50. [14]

ResNet101

- **Tasa de aprendizaje**: Se probaron valores entre 0.0001 y 0.2, siendo el valor de 0.0002 el más estable [Tabla 3].
- **Dropout:** Utilicé un dropout del 40%, para evitar el sobreajuste, pero manteniendo el equilibrio en la capacidad de aprendizaje del modelo.
- **Tamaño de batch:** Un batch de 64 ofreció el mejor equilibrio entre precisión y uso de recursos.
- **Número de épocas:** Se entrenó el modelo por 15 épocas para observar el comportamiento del aprendizaje.

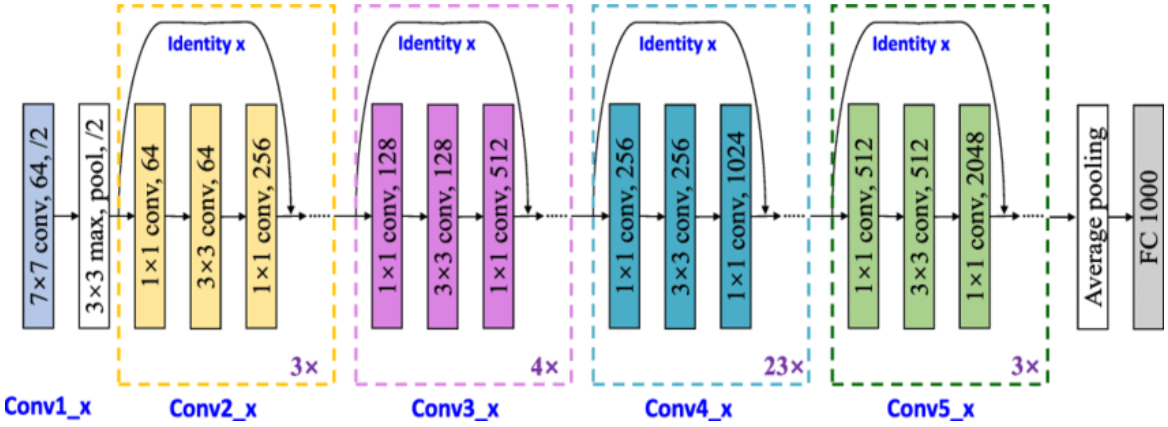

Figura 5: Representación visual de la arquitectura de la red neuronal ResNet 101. [15]

ResNet152

- **Tasa de aprendizaje:** Se probó una gama de tasas de aprendizaje, destacando 0.0005 como la más adecuada para la estabilidad [Tabla 4].
- **Dropout:** Utilicé un dropout del 40%, que funcionó bien para evitar el sobreajuste en esta red más profunda.
- **Tamaño de batch:** Los tamaños de batch probados variaron de 16 a 128, con el valor de 64 proporcionando el mejor resultado.
- **Número de épocas:** Se usaron 15 épocas para permitir que el modelo encontrara un buen ajuste a lo largo del tiempo.

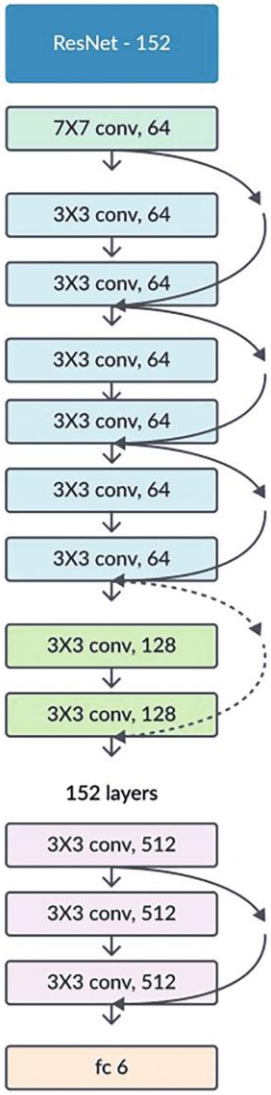

Figura 6: Representación visual de la arquitectura de la red neuronal ResNet 152. [16]

VGG16

- **Tasa de aprendizaje:** Configuré diferentes tasas, siendo el valor más efectivo de 0.0005, para obtener una precisión de validación del 58.33% [Tabla 5].
- **Dropout:** Utilicé valores de dropout del 20% y 40% para controlar el sobreajuste. La opción del 20% fue más efectiva en este modelo.
- **Tamaño de batch:** Experimenté con tamaños de batch desde 32 hasta 128. El valor óptimo fue 32, proporcionando el mejor balance entre estabilidad y tiempo de entrenamiento.
- **Número de épocas:** Utilicé hasta 15 épocas, lo que fue suficiente para observar cómo la precisión alcanzaba su punto máximo. De hecho, con tan sólo 5 épocas el modelo tiende a converger.

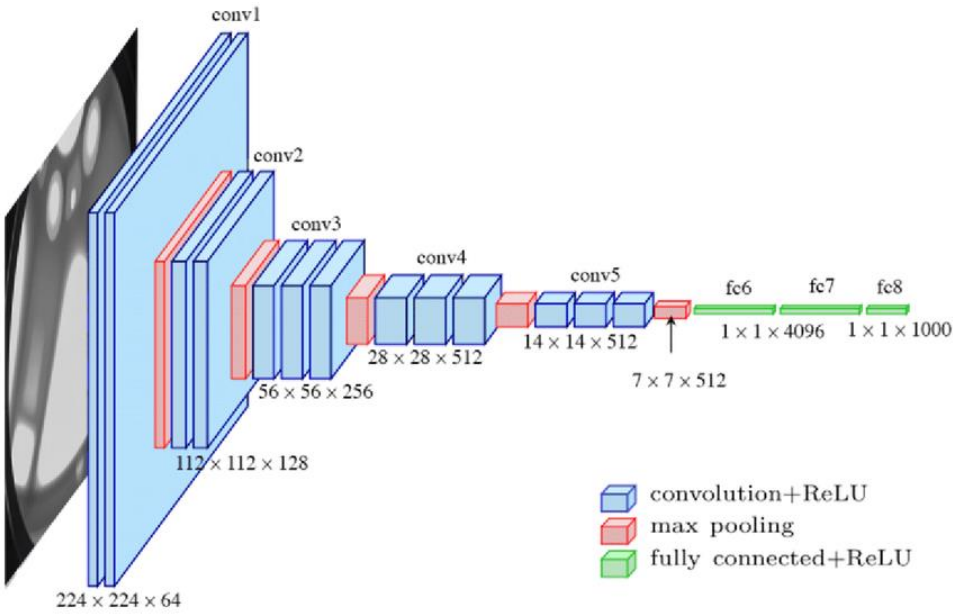

Figura 7: Representación visual de la arquitectura de la red neuronal VGG16. [17]

MobileNetV2 + Curriculum Learning

Para MobileNetV2 con Curriculum learning, configuré el entrenamiento de manera que el modelo se adaptara progresivamente a un conjunto de datos más complejo. Hay que mencionar que en todos los casos anteriores entrené el modelo con 10 razas, mientras que para el caso de MobileNetV2 + Curriculum Learning decidí utilizar 40 razas dado que el modelo tiene más margen de mejora al aplicar la acumulación de aprendizaje por cada fase puesto que puedo dividir 40 razas en 10 fases:

- **Tasa de aprendizaje:** La tasa de aprendizaje se mantuvo en 0.0004, proporcionando estabilidad y permitiendo un aprendizaje gradual.
- **Dropout:** Utilicé un dropout del 40% para mitigar el sobreajuste mientras se expandía el conjunto de datos en fases.
- **Tamaño de batch:** Se usó un batch size de 128 para aprovechar las capacidades computacionales sin comprometer la precisión.
- **Número de épocas:** Entrené el modelo durante 15 épocas en cada fase, permitiendo que aprendiera efectivamente en cada conjunto incremental de razas.

4.3 Aplicación de Curriculum Learning para Mejorar la Precisión del Modelo

4.3.1 Fundamentos de Curriculum Learning

Curriculum Learning es un enfoque de entrenamiento que presenta al modelo datos en etapas de dificultad creciente. En el caso de redes neuronales, se le muestra primero un conjunto de

datos sencillos y, gradualmente, uno más complejo. Esto permite que el modelo aprenda patrones básicos antes de enfrentar casos complejos, maximizando su capacidad de generalización.

4.3.2 Implementación de Curriculum Learning

1. **Definición de las Etapas de Entrenamiento:** Para implementar Curriculum Learning, se dividieron las razas de perros en subconjuntos organizados en niveles de dificultad. La idea es que el modelo aprenda progresivamente, comenzando con conjuntos pequeños y manejables, y aumentando la complejidad al añadir nuevas razas en cada fase. Así, el modelo adquiere una comprensión sólida de características básicas antes de abordar clasificaciones más complejas. [Apéndice F].

   Los subconjuntos de razas para cada fase se definieron de la siguiente manera:

   - **Fase 1 - Razas Iniciales:** Esta fase incluye razas con características visuales muy distintivas, facilitando que el modelo identifique patrones de forma sencilla sin analizar detalles complejos.
   - **Fase 2 - Ampliación de Complejidad Moderada:** Aquí se introducen razas con características algo más complejas, lo que supone un mayor reto de diferenciación de rasgos. Este subconjunto incluye razas de tamaño mediano a pequeño con variaciones en forma de cabeza, pelaje y proporciones.
   - **Fase 3 - Incremento en la Diversidad y Detalle de las Razas:** La tercera fase incluye razas que pueden parecer similares a las de fases anteriores, pero requieren que el modelo discrimine con mayor precisión texturas y estructuras. Esta fase abarca razas con variedad en estilo de pelaje, tamaños y detalles faciales, aumentando significativamente la complejidad de clasificación.

2. **Preprocesamiento y Configuración de Datos:** Para optimizar el entrenamiento, se desarrolló la función load_and_preprocess_images, que normaliza cada imagen en tamaño y valores de píxeles, garantizando así la homogeneidad del conjunto de datos. Las tareas clave incluyen:

   - **Redimensionamiento**: Todas las imágenes se ajustan a 224x224 píxeles, tamaño estándar en redes convolucionales como MobileNetV2, lo que facilita el procesamiento en lotes y evita variaciones de escala.
   - **Conversión a Array Numérico**: Cada imagen se transforma en un array de numpy, permitiendo que el modelo procese los datos como matrices.

- **Normalización de Píxeles**: Los valores de píxeles se normalizan a un rango [0, 1] al dividirlos entre 255, lo que estabiliza el aprendizaje y reduce problemas de gradiente.

3. **Entrenamiento Incremental con Validación Acumulativa:** En el entrenamiento incremental con validación acumulativa, el modelo se entrena en fases, donde en cada una de ellas se añade un nuevo conjunto de razas de perro a las razas ya aprendidas. Esto ayuda al modelo a adaptarse de manera progresiva a una mayor complejidad de datos, manteniendo al mismo tiempo la precisión sobre los datos previamente aprendidos.

    - **Proceso de Acumulación de Datos por Fase**
      Un nuevo conjunto de razas adicionales se combina con las razas de perros introducidas inicialmente. Este proceso de acumulación se implementa mediante un bucle en el que las razas de cada lista definida (fase_breeds_1, fase_breeds_2, etc.) se agregan de forma incremental. En cada iteración del bucle, se añaden nuevas razas al conjunto ya aprendido, lo cual permite al modelo adaptarse a los nuevos datos y, al mismo tiempo, retener el conocimiento de las razas previas.

    - **Establecimiento de la Validación Acumulativa**
      La validación acumulativa evalúa el rendimiento del modelo no solo en las razas de la fase actual, sino en todas las razas introducidas hasta ese momento. Al incluir los datos de todas las fases anteriores en la evaluación de la fase actual, se monitorea si el modelo mantiene la precisión sobre los datos previos mientras aprende nuevas razas. Esto es clave para detectar si el modelo comienza a "olvidar" conocimientos de etapas anteriores al incorporar nuevos datos, un fenómeno conocido como catástrofe de olvido.

4. **Ventajas de la Validación Acumulativa**

    - **Ventajas de la Validación Acumulativa**
      El enfoque de entrenamiento incremental con validación acumulativa favorece una evaluación continua y coherente del modelo en su proceso de aprendizaje. Este método permite el monitoreo progresivo del rendimiento. Esto, a su vez, permite observar si el modelo puede aprender nuevas razas manteniendo el conocimiento de las razas previamente aprendidas.

4.4 Evaluación y Optimización

4.4.1 Evaluación del Rendimiento

Cada modelo fue evaluado utilizando métricas de precisión de entrenamiento y precisión de validación, así como pérdida de entrenamiento y pérdida de validación. Estas métricas se calcularon al final de cada fase de entrenamiento, proporcionando información concisa sobre los resultados para poder sacar conclusiones precisas sobre la capacidad de aprender de cada modelo.

Todos los modelos, incluyendo VGG16, ResNet50, ResNet101, ResNet152 y MobileNetV2, fueron entrenados inicialmente con un conjunto de 10 razas de perros. Entre todos ellos, el modelo con mejor precisión de validación fue MobileNetV2, consiguiendo un 97.02%. Con esto se puede asumir que es la mejor arquitectura para la clasificación de razas de perros en este trabajo. Las arquitecturas de ResNet, especialmente las versiones más profundas (ResNet101 y ResNet152), mostraron resultados pobres en cuanto a precisión y pérdidas, probablemente debido a su complejidad y al número elevado de parámetros a ajustar.

4.4.2 Optimización del Modelo

Para optimizar el rendimiento, se probaron diversas configuraciones de hiperparámetros en cada modelo:

- **Tasa de aprendizaje:** Los valores moderados, desde 0.0001 a 0.0005, proporcionaron una convergencia estable, siendo estas tasas las que mejor rendimiento ofrecieron.
- **Dropout:** Se utilizó un valor de dropout del 0.4 en la mayoría de los modelos, lo cual fue efectivo para prevenir el sobreajuste en las redes más profundas, especialmente en MobileNetV2 y ResNet50.
- **Batch Size:** Los tamaños de batch variaron entre 16 y 128, siendo los valores de 64 y 128 los que mejor balance ofrecieron entre estabilidad y eficiencia computacional.
- **Número de Épocas:** Cada modelo fue entrenado con distintas épocas y aunque el modelo converge en 5-10 épocas, para learning rates pequeños, tarda hasta 15 épocas en converger. Estos tamaños de batch fueron suficiente para que el modelo alcanzara la convergencia sin caer en sobreentrenamiento.

4.4.3 Prueba adicional con MobileNetV2 y 40 razas usando Curriculum Learning

Después de observar el rendimiento destacado de MobileNetV2 con 10 razas, se realizó una prueba adicional utilizando un conjunto más desafiante de 40 razas. En esta prueba, se implementó Curriculum Learning en dos configuraciones: una con las 40 razas en una sola fase y otra dividida en 10 fases, añadiendo gradualmente grupos de razas.

Los resultados mostraron que, al entrenar con 40 razas en una sola fase, la precisión en validación alcanzó un 82.36%. Sin embargo, al aplicar Curriculum Learning con 10 fases, la precisión final en validación mejoró hasta un 87.00%, lo que representa un incremento del 5% en comparación con el enfoque de una sola fase. Este enfoque progresivo permitió al modelo adaptarse de manera gradual a la complejidad creciente del conjunto de datos, mejorando su capacidad para diferenciar entre razas similares.

5. Pruebas/Experimentos

Para las siguientes pruebas han usado los siguientes rangos de valores:

- Learning rate: [0.0001 - 0.5]
- Dropout: [0 - 0.4]
- Batch size: [16 - 128]
- Número de razas: 10

Con estos rangos de valores, se han creado las tablas que se verán a continuación con su correspondiente comparación y análisis.

Las pruebas se han realizado haciendo un barrido entre todos los valores de cada hiperparámetro, encontrando de esta manera el top 10 que mejor precisión y pérdidas nos ofrece. Tras procesar las imágenes y elegir el modelo de arquitectura, entrenamos dicho modelo con la mejor combinación y lo usamos para predecir nuevas imágenes.

5.1. CNN Simple

La arquitectura de red neuronal convolucional (CNN) está diseñada para ser simple y eficiente, enfocándose en una estructura jerárquica que permite extraer características relevantes de las imágenes. Aunque no emplea técnicas avanzadas como las convoluciones separables en profundidad, su diseño modular asegura un buen rendimiento en tareas de clasificación.

Arquitectura: La arquitectura personalizada de red neuronal convolucional (CNN) sigue un diseño clásico, combinando capas convolucionales y de pooling para extraer y reducir características de manera eficiente. Consta de cuatro bloques convolucionales con un número creciente de filtros (32, 64, 128 y 256), seguidos de operaciones de Max Pooling que reducen la dimensionalidad espacial. Antes de la salida, se incluye una capa de Dropout con una tasa del 40% para prevenir el sobreajuste. Finalmente, una capa completamente conectada con activación softmax genera las probabilidades de clasificación para cada clase objetivo

5.2. MobileNetV2

MobileNetV2 es una arquitectura ligera diseñada específicamente para dispositivos con recursos limitados, como teléfonos móviles. La clave de su eficiencia radica en el uso de convoluciones separables en profundidad y bloques residuales invertidos. Estas técnicas permiten reducir drásticamente el número de operaciones necesarias, lo que facilita su implementación en aplicaciones en tiempo real.

Arquitectura: MobileNetV2 utiliza convoluciones separables en profundidad, que dividen la operación de convolución en dos fases: la convolución en profundidad y la convolución puntual (1x1). Esta separación reduce significativamente la carga computacional.

Tabla 1: Representación de los diez mejores resultados de precisión para la arquitectura MobileNetV2.

TOP	learning_rate	dropout	batch_size	train_accuracy	val_accuracy	train_loss	val_loss	time_elapsed_sec
1	0,0005	0,2	32	1,0000	0,9702	0,0000	0,37	229,9
2	0,5	0,4	64	0,9985	0,9702	0,0539	222,26	211,4
3	0,0002	0,4	64	1,0000	0,9643	0,0003	0,17	204,8
4	0,0002	0,4	128	1,0000	0,9643	0,0005	0,14	210,9
5	0,005	0,4	64	1,0000	0,9643	0,0000	1,99	223,0
6	0,01	0,4	128	1,0000	0,9643	0,0000	5,14	197,2
7	0,2	0,4	64	1,0000	0,9643	0,0000	60,96	209,5
8	0,0001	0,2	64	1,0000	0,9583	0,0014	0,20	398,3
9	0,0001	0,4	32	1,0000	0,9583	0,0043	0,15	469,0
10	0,0005	0	16	1,0000	0,9583	0,0000	0,38	277,7

Rendimiento: En los experimentos, MobileNetV2 ha mostrado consistentemente altas tasas de precisión de validación, alcanzando hasta un 97.02% con un learning rate de 0.0005, un dropout de 0.2, y un batch size de 32. Estos resultados demuestran su eficiencia tanto en términos de velocidad como de precisión, haciéndola una opción ideal para aplicaciones donde los recursos son limitados, como sistemas embebidos.

Figura 8: Imagen utilizada para la predicción dentro del modelo.

Para la arquitectura MobileNetV2, con la cual se ha obtenido el mejor rendimiento en cuanto a precisión, tiempo y recursos, se ha realizado una representación de activación neuronal con la siguiente imagen de la raza great Dane.

En las Figuras 9, 10 y 11, mostradas a continuación, con la imagen de la Figura 8, se muestran la activación neuronal de las capas "block_1_expand_relu", "block_5_expand_relu" y "block_15_expand_relu", apreciando así los patrones aprendidos por las neuronas en cada capa.

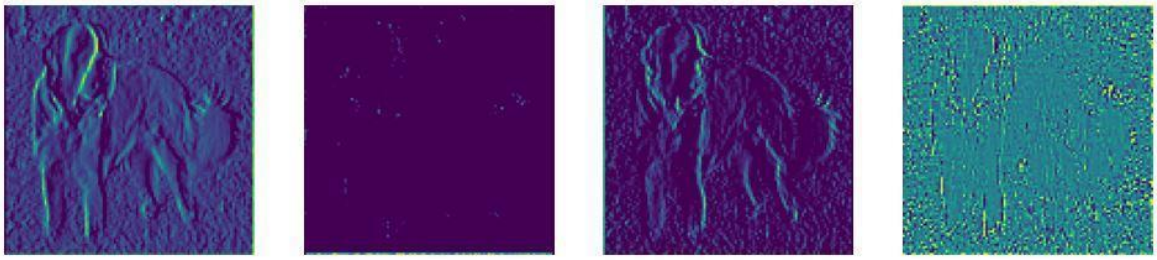

Figura 9: Representación de la activación neuronal en la capa "block_1_expand_relu".

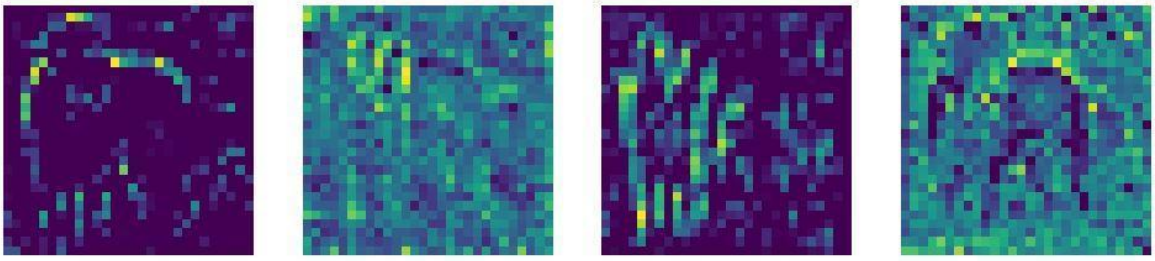

Figura 10: Representación de la activación neuronal en la capa "block_5_expand_relu".

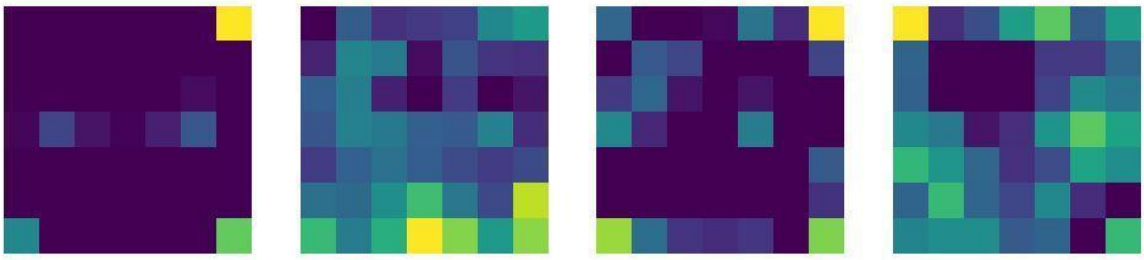

Figura 11: Representación de la activación neuronal en la capa "block_15_expand_relu".

MobileNetV2 + Curriculum Learning

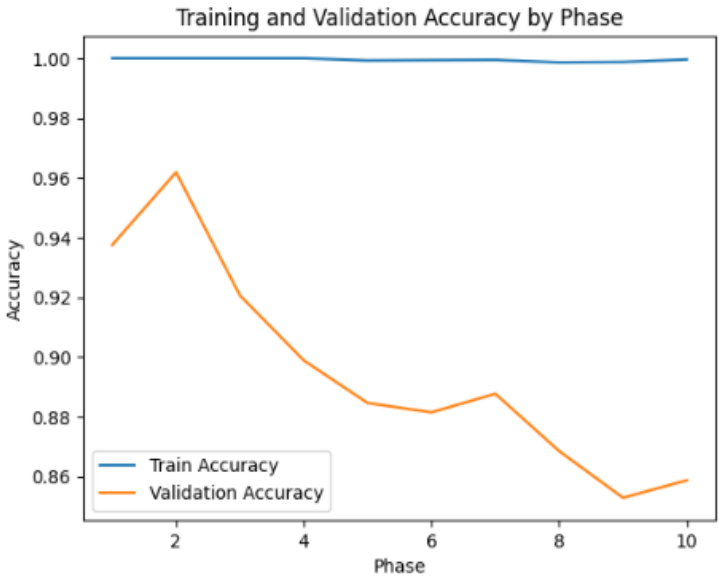

Figura 12: Representación de la precisión de entrenamiento del modelo MobileNetV2 frente a la de validación por fase con 40 razas.

La precisión de entrenamiento consigue el 100% en apenas unas épocas mientras que la precisión máxima de validación se sitúa en un 97% en la primera fase y 87% en la décima y última fase.

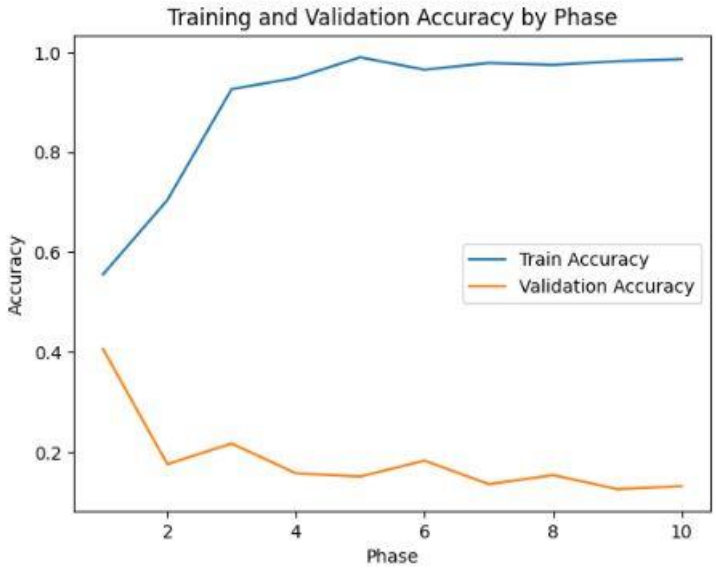

Figura 13: Representación de la precisión de entrenamiento del modelo CNN frente a la de validación por fase con 40 razas.

Tras comparar la Figura 12 con la Figura 13, se observa cómo el rendimiento, tras usar Curriculum Learning en ambos modelos, sigue siendo altamente dispar. Esto es debido a que el modelo MobileNetV2 se adapta perfectamente a nuestros datos, mientras que la CNN simple desde un primer momento, incluso sin Curriculum Learning ha dado resultados negativos.

5.3. ResNet (50, 101, 152)

Las redes ResNet introducen conexiones residuales, que permiten el entrenamiento de redes mucho más profundas sin sufrir problemas de vanishing gradient (gradiente que se desvanece). Las variantes ResNet50, ResNet101 y ResNet152 se diferencian principalmente en la cantidad de capas y bloques residuales, lo que les permite capturar características más complejas en los datos.

Arquitectura: ResNet50, con 50 capas, es la más ligera entre las variantes evaluadas, mientras que ResNet152, con 152 capas, es la más profunda. Estas arquitecturas permiten una gran capacidad de representación, aunque a costa de un mayor tiempo de entrenamiento.

Tabla 2: Representación de los diez mejores resultados de precisión para la arquitectura ResNet 50.

TOP	learning_rate	dropout	batch_size	train_accuracy	val_accuracy	train_loss	val_loss	time_elapsed_sec
1	0,0005	0	16	1,0000	0,5774	0,0323	1,3585	3430,34
2	0,2	0,4	64	0,9955	0,5655	0,1205	79,2049	7361,30
3	0,0002	0,4	64	0,8826	0,5536	0,5963	1,5335	3430,71
4	0,005	0,4	64	0,9985	0,5536	0,0087	2,0312	3653,22
5	0,0005	0,2	32	1,0000	0,5476	0,0762	1,3685	3347,13
6	0,5	0,4	64	0,9881	0,5119	1,0809	223,2146	3386,86
7	0,01	0,4	128	0,9629	0,5119	0,2167	5,1247	3616,57
8	0,0001	0,4	32	0,7905	0,5000	0,8738	1,5552	3450,48
9	0,0001	0,2	64	0,8366	0,4940	0,8921	1,6589	3362,51
10	0,0002	0,4	128	0,8143	0,4643	0,8221	1,6216	3559,37

Tabla 3: Representación de los diez mejores resultados de precisión para la arquitectura ResNet 101.

Top	learning_rate	dropout	batch_size	train_accuracy	val_accuracy	train_loss	val_loss	time_elapsed_sec
1	0,0002	0,4	64	0,2957	0,2321	2,03	2,21	1332,55
2	0,0001	0,2	64	0,2496	0,2202	2,07	2,22	1380,37
3	0,01	0,4	128	0,3789	0,2083	6,88	9,88	1270,92
4	0,5	0,4	64	0,2704	0,2024	683,68	764,32	1399,71
5	0,005	0,4	64	0,4279	0,2024	2,89	5,24	1291,04
6	0,0001	0,4	32	0,2110	0,1905	2,30	2,17	1392,05
7	0,0005	0,2	32	0,4012	0,1667	1,83	3,23	1357,54
8	0,0005	0	16	0,4933	0,1607	1,54	3,08	1395,05
9	0,0002	0,4	128	0,2615	0,1429	2,20	2,22	1337,01
10	0,2	0,4	64	0,3195	0,1310	221,12	299,72	1342,05

Tabla 4: Representación de los diez mejores resultados de precisión para la arquitectura Resnet152.

Top	learning_rate	dropout	batch_size	train_accuracy	val_accuracy	train_loss	val_loss	time_elapsed_sec
1	0,0001	0,2	64	0,27192	0,2500	2,06	2,18	1963,66
2	0,2	0,4	64	0,35513	0,2024	190,32	223,59	1951,41
3	0,0002	0,4	128	0,27043	0,1964	2,19	2,23	1906,76
4	0,01	0,4	128	0,34175	0,1905	10,64	14,73	1916,30
5	0,0001	0,4	32	0,24071	0,1845	2,26	2,39	1992,77
6	0,0005	0	16	0,53195	0,1845	1,46	2,73	2009,90
7	0,0002	0,4	64	0,28083	0,1726	2,10	2,30	1933,15
8	0,5	0,4	64	0,34473	0,1607	460,79	964,89	1928,91
9	0,005	0,4	64	0,36999	0,1250	4,35	8,98	1928,56
10	0,0005	0,2	32	0,46211	0,0952	1,70	3,25	1992,51

Rendimiento: En los experimentos, ResNet50 alcanzó una precisión de validación de hasta 55.95% con un learning rate de 0.0005 y un batch size de 64, mientras que ResNet152 mostró un rendimiento máximo con una precisión de validación de 25%. Estos resultados indican que, aunque las redes más profundas tienen un potencial teórico superior, su rendimiento puede verse afectado si los datos no justifican una gran capacidad de representación.

5.4. VGG16

VGG16 es una red profunda, simple en su diseño, caracterizada por el uso de bloques convolucionales repetitivos con capas de pooling que reducen progresivamente la resolución espacial. Aunque es una de las arquitecturas más utilizadas para la clasificación de imágenes, su principal desventaja es su elevado número de parámetros, lo que conlleva un costo computacional significativo.

Arquitectura: VGG16 utiliza convoluciones de 3x3 con max-pooling después de cada conjunto de capas convolucionales, seguido de capas completamente conectadas al final. Esta estructura facilita la detección de características complejas, pero a un alto costo computacional.

Tabla 5: Representación de los diez mejores resultados de precisión para la arquitectura VGG16.

TOP	learning_rate	dropout	batch_size	train_accuracy	val_accuracy	train_loss	val_loss	time_elapsed_sec
1	0,0005	0,2	32	1,0000	0,5833	0,08	1,34	3397,86
2	0,005	0,4	64	1,0000	0,5774	0,00	2,15	4195,82
3	0,5	0,4	64	0,9866	0,5595	1,90	225,37	3123,49
4	0,0002	0,4	64	0,9123	0,5595	0,57	1,48	3115,24
5	0,01	0,4	128	0,9718	0,5417	0,11	5,25	3711,07
6	0,0005	0	16	1,0000	0,5417	0,04	1,38	3438,67
7	0,0001	0,4	32	0,8009	0,5238	0,85	1,64	3424,27
8	0,2	0,4	64	0,9926	0,5060	0,33	90,79	3457,34
9	0,0002	0,4	128	0,7979	0,5000	0,87	1,56	3144,28
10	0,0001	0,2	64	0,8187	0,4702	0,90	1,68	7951,97

Rendimiento: VGG16 mostró un rendimiento un tanto pobre, alcanzando una precisión de validación de 58.33% en los mejores experimentos. Además, debido a su estructura pesada, también fue la arquitectura que más tiempo tomó entrenar, lo que podría no ser ideal en escenarios donde los recursos o el tiempo son limitados.

6. Resultados

6.1. Evaluación de Hiperparámetros y Técnicas de Regularización

6.1.1 Dropout

Dropout es una técnica de regularización que previene el sobreajuste al desactivar aleatoriamente un conjunto de neuronas durante el entrenamiento. Esto fuerza a la red a aprender representaciones más robustas que son menos dependientes de unidades específicas.

Impacto en el Rendimiento: En las pruebas, se observó que un Dropout moderado (alrededor de 0.2 a 0.4) era beneficioso en la mayoría de las arquitecturas, ayudando a prevenir el sobreajuste. Por ejemplo, en MobileNetV2, un Dropout de 0.2 con un learning rate de 0.0005 resultó en una alta precisión de validación del 97.02%. Por otro lado, Dropout demasiado alto podría llevar a una pérdida de información relevante, como se observó en algunos casos de ResNet donde la precisión de validación cayó notoriamente.

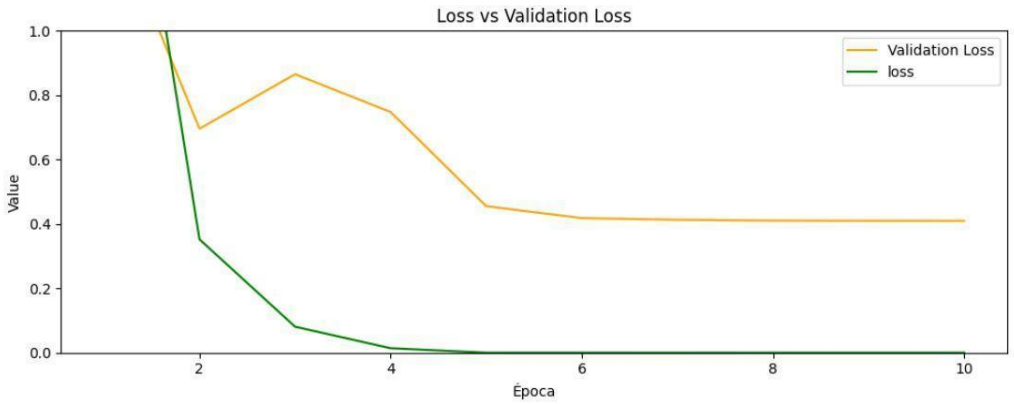

Figura 14: Representación de las pérdidas de validación frente a la pérdida de entrenamiento usando MobileNetV2 con 10 razas.

A medida que se realizan épocas, las pérdidas de entrenamiento bajan hasta 0 de manera exponencial dado que su precisión alcanza el 100%, mientras que las pérdidas de validación convergen en 0.4 por la misma razón. [Figura 14]

6.1.2. Learning Rate

La tasa de aprendizaje es un factor clave en el entrenamiento de modelos de aprendizaje automático, ya que determina el tamaño de las actualizaciones que se realizan en los pesos del

modelo en cada iteración. Un valor de tasa de aprendizaje demasiado alto puede resultar en una disminución de aprendizaje del modelo, dificultando su capacidad para converger a una solución óptima. Por el contrario, una tasa de aprendizaje demasiado baja puede hacer que el modelo converja de manera excesivamente lenta o que quede atrapado en mínimos locales, lo que perjudica su rendimiento.

Ajuste Fino (Fine-Tuning): Al analizar los experimentos realizados con diferentes tasas de aprendizaje, se observó que valores desde 0.0005 al 0.001 generalmente ofrecieron los mejores resultados a través de diversas arquitecturas. Por ejemplo, en el modelo VGG16, una tasa de 0.0005 logró una precisión de validación de 58%, mientras que en ResNet101 y ResNet50, tasas más pequeñas también demostraron un rendimiento efectivo. Esto es coherente con la práctica común de usar tasas de aprendizaje más bajas en modelos más complejos como ResNet y VGG16, que requieren un ajuste más delicado debido a su alta capacidad de representación.

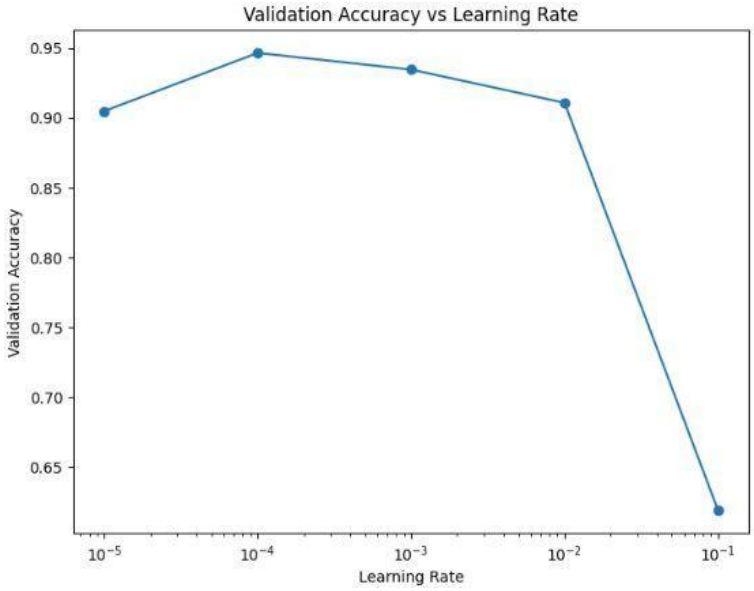

Figura 15: Representación de la precisión de validación en función del learning rate empleado usando MobileNetV2 con 10 razas.

6.1.3. Batch Size

El tamaño de lote determina cuántos ejemplos se procesan antes de que el modelo actualice sus pesos. Utilizar un tamaño de lote mayor generalmente proporciona estimaciones más estables del gradiente, pero también requiere más memoria y puede ser menos efectivo en términos de generalización.

Se encontraron tamaños de lote moderados, como 64 y 128, efectivos en las arquitecturas evaluadas. Por ejemplo, en el modelo MobileNetV2, un tamaño de lote de 32 dio lugar a una precisión de validación de 97%, mientras que en ResNet50, un tamaño de lote de 64 resultó en

una precisión de 55%. Sin embargo, en este modelo, no afecta tanto el batch size, sino el learning rate. Se aprecia que el learning rate bajo contribuye a una mejor precisión mientras que los valores de batch size se utilizan para un ajuste fino pudiendo variar entre 1 y 3 puntos la precisión del modelo, siempre y cuando los valores del batch size no sean extremadamente pequeños ni, en su contrario, extremadamente grandes. [16]

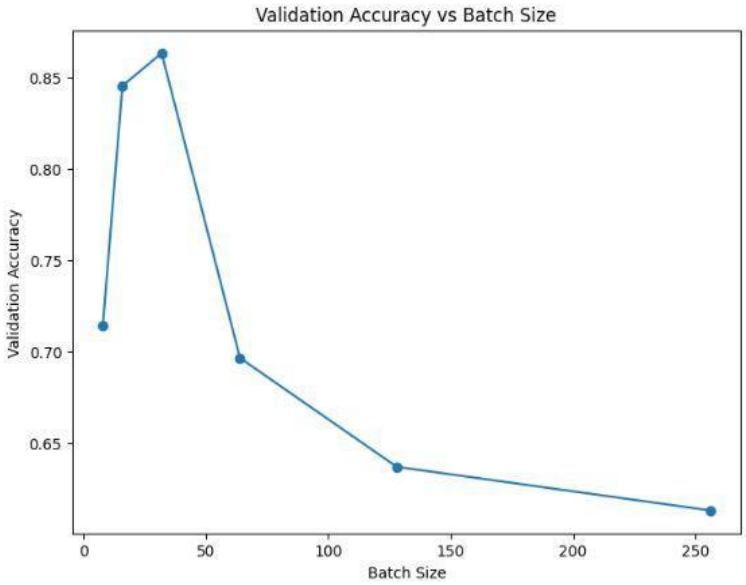

Figura 16: Representación de la precisión de validación en función del tamaño de batch size.

6.1.4. Número de Neuronas

El número de neuronas en una capa densa tiene un impacto significativo en la capacidad de aprendizaje de un modelo. Si bien un mayor número de neuronas puede capturar patrones más complejos, también incrementa el riesgo de sobreajuste y el costo computacional.

Las evaluaciones realizadas revelaron que un rango moderado de neuronas, entre 32 y 64, es particularmente efectivo. Por ejemplo, un modelo con 64 neuronas logró una precisión de validación cercana al 91%. En contraste, al utilizar 128 o más neuronas, la precisión disminuyó levemente, lo que sugiere la posibilidad de sobreajuste o redundancia en la representación.

La gráfica [Figura 17] demuestra que un aumento inicial en el número de neuronas mejora considerablemente la precisión, pero este efecto se estabiliza a partir de 64 neuronas. A partir de este punto, los incrementos son marginales o incluso se producen ligeros descensos. Esto indica que el número óptimo de neuronas no siempre es el mayor, sino aquel que equilibra de manera adecuada la capacidad de aprendizaje y la generalización del modelo.

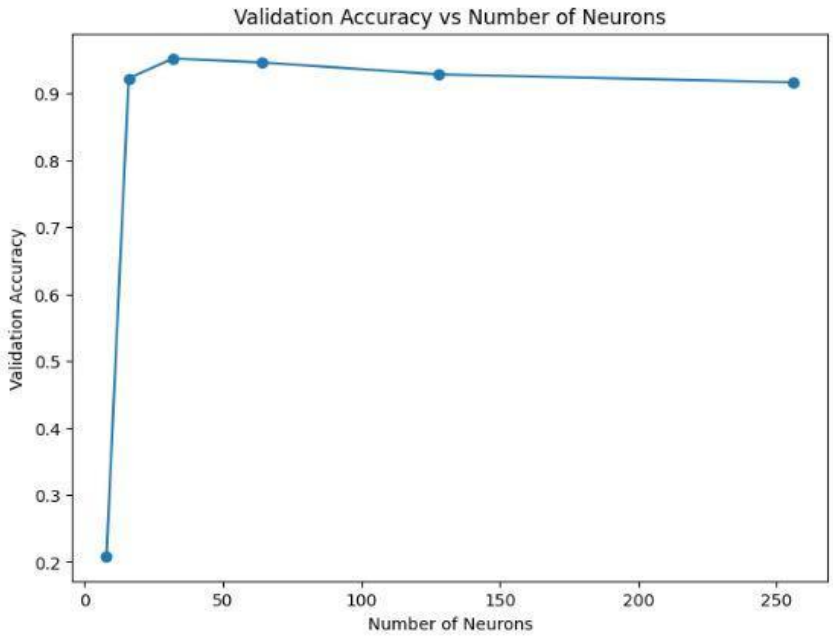

Figura 17: Representación de la precisión de validación en función del número de neuronas en su última capa.

6.2. Análisis de las Arquitecturas

6.2.1. CNN Simple

El modelo propuesto, una arquitectura de red convolucional propia, presenta un diseño sencillo y efectivo basado en capas convolucionales con activaciones ReLU y operaciones de pooling para reducir la dimensionalidad. Este enfoque estructurado permite extraer características visuales progresivamente más complejas a través de diferentes niveles de la red, mientras que la inclusión de una capa de Dropout con una tasa de 40% contribuye a mitigar el sobreajuste.

Sin embargo, los resultados obtenidos muestran ciertas limitaciones en cuanto a la capacidad de generalización del modelo. Con una tasa de aprendizaje de 0.0005, un dropout de 0% y un tamaño de batch de 16, se logró una precisión de validación de 23.81%. Por otro lado, incrementando el dropout al 40% y utilizando un tamaño de batch de 64, se alcanzó una precisión ligeramente superior de 27.14%.

6.2.2. MobileNetV2

El punto destacado de MobileNetV2 es su enfoque en la eficiencia computacional, utilizando convoluciones separables en profundidad, que son mucho más eficientes y reducen los parámetros en gran medida en comparación con arquitecturas más clásicas como VGG16 o ResNet. Este diseño optimizado lo convierte en una de las mejores opciones para dispositivos con recursos limitados. MobileNetV2 alcanzó un rendimiento sobresaliente con una precisión

del 97.02% en el conjunto de validación y, además, tomó mucho menos tiempo de entrenamiento en comparación con arquitecturas de configuraciones mucho más complejas como ResNet152 o VGG16, debido a la cantidad de parámetros. La conclusión presentada indica que MobileNetV2 es una arquitectura balanceada que consume recursos de manera eficiente, siendo una excelente solución para aplicaciones en tiempo real que exigen modelos rápidos sin comprometer demasiado la precisión alcanzada.

6.2.3. ResNet (50, 101, 152)

ResNet, con la innovación de conexiones residuales, permite entrenar redes neuronales profundas sin sufrir el problema del vanishing gradient. Sin embargo, la relación entre la profundidad del modelo y el rendimiento en este conjunto de datos indica que mayor profundidad no siempre se traduce en mejor precisión. ResNet50 alcanzó una tasa de precisión del 55.95%, mientras que las versiones más profundas, ResNet101 y ResNet152, que no mostraron grandes mejoras, obtuvieron precisiones de 25% y 20%, respectivamente.

Tiempo de entrenamiento: El tiempo de entrenamiento aumentó con la profundidad de la red, siendo ResNet152 la más costosa en términos de tiempo y uso de memoria. Este incremento en la complejidad no parece compensarse con suficientes beneficios para equilibrar el incremento de recursos necesarios para su entrenamiento.

Conclusión: ResNet50 parece ser una opción más equilibrada, ofreciendo una precisión moderada en combinación con un tiempo de entrenamiento razonablemente reducido. Profundizar la arquitectura no resultó ventajoso para este conjunto de datos en particular.

6.2.4. VGG16

VGG16 es estructuralmente simple debido a la repetición de bloques convolucionales y max-pooling, lo que la convierte en una arquitectura costosa en términos de computación y parámetros.

Tasa de precisión: En los experimentos, VGG16 alcanzó una precisión del 58.33%, situándose por debajo de MobileNetV2, pero por encima de ResNet50.

Tiempo de entrenamiento: VGG16 tuvo un costo computacional significativamente alto en comparación con MobileNetV2 y ResNet50, lo que reduce su idoneidad para entornos con limitaciones de recursos.

Conclusión: En este sentido, VGG16, siendo más rigurosa y efectiva, es relevante cuando la precisión es la prioridad y los recursos no son una limitación crítica. Sin embargo, su alto costo computacional reduce considerablemente su aplicabilidad en situaciones donde la eficiencia es una preocupación importante.

6.3. Interpretación de los Resultados

De los experimentos realizados con diferentes arquitecturas de redes neuronales convolucionales se pueden deducir las siguientes conclusiones:

Eficiencia de MobileNetV2: MobileNetV2 destaca al alcanzar una precisión del 97.02% y lo hace con tiempos de entrenamiento reducidos, demandando pocos recursos. Esto la hace ideal para aplicaciones que requieren un buen balance entre velocidad y rendimiento, como en dispositivos móviles o sistemas embebidos.

Complejidad innecesaria en redes más profundas de ResNet: Las versiones profundas de ResNet (101 y 152) no lograron mejoras significativas en precisión, lo cual podría deberse a la naturaleza del dataset o a un posible sobreajuste del modelo. ResNet50 mostró un rendimiento competitivo con menos capas, logrando un equilibrio adecuado entre capacidad de representación y tiempo de entrenamiento.

VGG16: Precisión frente al costo computacional VGG16 permite alcanzar una precisión aceptable, pero su gran número de parámetros la hace poco práctica para situaciones con limitaciones de recursos. Está diseñada para aplicaciones que requieren una precisión extremadamente alta, aunque en otros casos el tiempo de entrenamiento y el consumo de memoria pueden resultar problemáticos.

Adecuación al Dataset: Aunque el dataset completo contiene 120 razas, se realizaron pruebas iniciales con solo 10 razas para evaluar el desempeño de cada modelo. Al utilizar únicamente 10 razas, utilizar redes muy profundas como ResNet101 o ResNet152 podría ser excesivo. Modelos más ligeros como MobileNetV2 o versiones simples de ResNet son más efectivos en este contexto.
Posteriormente, se aplica la técnica de currículum learning con 40 razas. Con este número de razas, se elige utilizar Curriculum learning para introducir los datos del dataset de manera gradual, beneficiando el aprendizaje del modelo.

Ajuste de Hiperparámetros: Los hiperparámetros más conservadores fueron un learning rate bajo de 0.0005 y un batch size moderado de 64. Además, el uso de dropout fue clave para evitar el sobreajuste, especialmente en redes profundas como VGG16 y ResNet.

6.4 Comparación de Resultados y Mejora en Precisión

La comparación final muestra que el modelo entrenado con Curriculum Learning logra una mejora en precisión de un 5% en la clasificación de 40 razas de perros en comparación con el modelo estándar, en el cual se obtuvo, con, MobileNetV2 y 40 razas, una precisión de 82%. Además, el Curriculum Learning no solo incrementa la precisión, sino también la estabilidad del aprendizaje, permitiendo que el modelo refuerce patrones en etapas antes de enfrentarse a la clasificación completa de razas.

En conclusión, MobileNetV2 se presenta como la mejor opción para este proyecto por su equilibrio entre precisión y eficiencia. El modelo propio, ResNet50, 101, 152 y VGG16, presentan resultados poco precisos pues no se consigue superar la precisión de 60% por lo que no serían alternativas reales. No sólo eso, sino que en los tiempos de entrenamientos también se aprecia una diferencia demasiado grande como para tomar cualquiera de estas opciones como alternativa de la MobileNetV2. Para ser más concretos, la MobileNetV2 tarda entre 200 y 300 segundos para 10 épocas y 10 razas, mientras que el modelo más próximo en cuanto a tiempo se trata de la ResNet101, con un tiempo de entrenamiento de 1300-1400 segundos, es decir hasta 6 veces más. El modelo con mayor tiempo de entrenamiento sería e VGG16 con tiempos de 3000-8000 segundos.

7. Conclusiones y Trabajo Futuro

7.1 Conclusiones de las Diferentes Arquitecturas de Red Neuronal

CNN Simple: Este modelo dio unos resultados pobres, no superando el 30% de precisión de validación. Por otro lado, fue el segundo modelo con mejor tiempo de entrenamiento y de coste computacional. Esto se debe a la arquitectura simple con pocas capas y estructura de Max Pooling y Conversión 2D principalmente. No es el modelo óptimo, pero si es mas recomendable que la ResNet152 dado que obtienen precisiones similares pero el tiempo de entrenamiento y su coste computacional es notoriamente inferior.

MobileNetV2: Este modelo mostró un rendimiento casi constante y sobresaliente en términos de precisión de validación, alcanzando casi un 97% en su configuración óptima. Esto evidencia que las convoluciones separables en profundidad y los bloques residuales invertidos son efectivos para mantener un alto rendimiento con un bajo costo computacional.

ResNet: Aunque ResNet50 logró una precisión bastante buena, las versiones más profundas (ResNet101 y ResNet152) no consiguieron un rendimiento significativamente mejor. Esto podría indicar que el dataset no justifica el aumento de complejidad de estos modelos más profundos o que los hiperparámetros requieren un ajuste más preciso.

VGG16: Aunque VGG16 es capaz de capturar características muy complejas, su gran cantidad de parámetros y su extenso tiempo de entrenamiento la hacen inapropiada para situaciones en las que la eficiencia es crucial. Sin embargo, sigue siendo una opción viable cuando la precisión es la prioridad y la disponibilidad de recursos no es un problema.

Al comparar los resultados de este estudio con la literatura reciente, MobileNetV2 se destaca tanto en los experimentos realizados como en los estudios de Raduly et al. (2018) [8], gracias a su alta precisión, bajo costo computacional y rapidez en el entrenamiento, consolidándose como una opción eficiente y efectiva. Por otro lado, modelos como ResNet50, ResNet101 y ResNet152 no mostraron buenos resultados en este análisis, lo que sugiere la necesidad de ajustar su complejidad al dataset tal y como Gunaranjan et al. (2024) [10] demostraron con ajustes adecuados. Además, Saglietti et al. (2022) [11] enfatizaron que el Curriculum Learning mejora la generalización y acelera el entrenamiento; en nuestro análisis, su implementación permitió aumentar la precisión del 82% al 87%, destacando su valor en tareas con datos complejos y en escenarios donde optimizar el rendimiento es fundamental. Por último, aunque VGG16 no obtuvo buenos resultados en este estudio, podría considerarse viable en contextos donde los recursos no son una limitación y se prioriza exclusivamente la precisión.

7.2. Consideraciones sobre Preprocesamiento de Imágenes

El preprocesamiento de imágenes es un paso esencial en la preparación de datos para redes neuronales. La normalización de los valores de píxeles y el redimensionamiento son cruciales para garantizar que la red pueda aprender de manera eficiente.

Normalización: La normalización de todas las imágenes en el rango [0, 1] ayudó a estabilizar el entrenamiento, evitando gradientes grandes que podrían causar una convergencia inestable.

Redimensionamiento: Las imágenes fueron redimensionadas a un tamaño estándar (224x224) compatible con las entradas esperadas por las redes preentrenadas, lo que asegura compatibilidad con la arquitectura de la red y previene errores en la inferencia.

7.3. Trabajo futuro

En aplicaciones reales, la elección de la arquitectura debe estar alineada con las limitaciones de recursos y los requisitos de precisión. En este caso, MobileNetV2 sería la mejor opción para aplicaciones móviles o dispositivos embebidos, para tareas que requieren un equilibrio entre precisión y capacidad computacional. VGG16, con su alto consumo de recursos, no podría sería apta para trabajos en aplicaciones donde la precisión extrema es crítica y los recursos no son una limitación.

7.3.1. Exploración de Optimizadores Avanzados

En los experimentos realizados, se utilizó principalmente el optimizador Adam. Este es una extensión de SGD con momento y una adaptación de la tasa de aprendizaje basada en los primeras y segundas derivadas de los gradientes. Aunque Adam es altamente efectivo y ampliamente usado por su capacidad para manejar problemas de aprendizaje profundo con tasas de aprendizaje fluctuantes durante el entrenamiento, existen otros optimizadores avanzados que podrían explorarse para obtener mejores resultados.

7.3.2. Augmentación de Datos

La augmentación de datos es una técnica muy importante para mejorar la capacidad de generalización de cualquier modelo de aprendizaje profundo. Aunque en los experimentos actuales se aplicaron augmentaciones básicas, como rotaciones, traslaciones, explorar técnicas más avanzadas en esta área podría aportar mejoras en el rendimiento.

7.3.3. Reducción de la Complejidad en Modelos Complejos

Las arquitecturas como ResNet152 y VGG16, aunque poderosas, tienen un alto costo computacional y son propensas al sobreajuste si no se dispone de una cantidad suficiente de datos de entrenamiento o si no se ajustan correctamente los hiperparámetros.

7.3.4. Estrategias a Considerar

Pruning: Esta técnica implica eliminar pesos innecesarios o irrelevantes en la red después del entrenamiento inicial, lo que permite reducir tanto el tamaño del modelo como el tiempo de inferencia, sin afectar notablemente la precisión.

Ensamble: El ensamble en redes neuronales es un método donde se combinan varios modelos para mejorar el rendimiento general en tareas como clasificación o regresión. La idea central es que, al combinar múltiples modelos, las debilidades individuales pueden mitigarse, mientras que la precisión y robustez del sistema en su conjunto aumentan considerablemente.

Conceptos Clave del Ensamble:

- **Diversidad de Modelos**: Los modelos en un ensamble pueden ser del mismo tipo (por ejemplo, varias redes neuronales) o de tipos distintos (como una mezcla de redes neuronales y árboles de decisión). La diversidad es importante porque diferentes modelos pueden captar distintos aspectos de los datos.

Métodos de Combinación

- **Voting**: En clasificación, los modelos "votan" y la clase final se decide por mayoría; en regresión, las predicciones se promedian.
- **Stacking**: En este método, las salidas de los modelos base se usan como entrada para un modelo final que aprende cómo combinar estas predicciones.
- **Bagging**: Se entrena a varios modelos en diferentes subconjuntos de datos, generalmente usando muestreo con reemplazo.

7.3.5. Exploración de Arquitecturas Alternativas

Si bien MobileNetV2, ResNet y VGG16 son arquitecturas consolidadas, el campo de la visión por computadora sigue evolucionando, con nuevas arquitecturas emergentes que podrían mejorar el rendimiento y la eficiencia.

Arquitecturas a Considerar

- **EfficientNet**: Esta familia de modelos escala de forma eficiente la profundidad, el ancho y la resolución de entrada, logrando un rendimiento superior con menos parámetros y operaciones, optimizando tanto precisión como eficiencia.
- **Vision Transformers (ViT)**: Aunque inicialmente fueron creados para tareas de procesamiento de lenguaje natural, los transformers están siendo rápidamente adaptados para visión por computadora. Ofrecen una nueva manera de abordar la estructura de las imágenes, capturando dependencias a largo plazo en los datos y proporcionando una perspectiva innovadora para el manejo de imágenes.

7.4. Conclusión final

Los experimentos realizados con MobileNetV2, ResNet y VGG16 revelan un panorama diverso en cuanto a rendimiento y eficiencia. MobileNetV2, con su diseño optimizado, destaca en aplicaciones con recursos limitados, mientras que ResNet y VGG16 ofrecen alternativas potentes para casos que demandan una mayor capacidad de representación. Sin embargo, las proyecciones futuras apuntan a mejoras significativas en precisión y eficiencia, especialmente mediante la adopción de nuevas técnicas de optimización, augmentación de datos y exploración de arquitecturas emergentes.

Este análisis indica que, para lograr el máximo rendimiento en aplicaciones prácticas, es clave no solo elegir la arquitectura adecuada, sino también ajustar los hiperparámetros cuidadosamente y tener en cuenta las limitaciones del hardware donde se implementarán los modelos. Al combinar estas estrategias, se pueden maximizar tanto la precisión como la eficiencia, impulsando aplicaciones de aprendizaje profundo más efectivas y accesibles.

Bibliografía

[1] Kaggle. (n.d.). *Dog breed identification*. Kaggle. Recuperado de https://www.kaggle.com/search?q=dog+breed+identification

[2] Hernández, S. (2022.). *Machine Learning desde cero*. Udemy. Recuperado de https://www.udemy.com/course/machine-learning-desde-cero/?couponCode=JUST4U02223

[3] SuperDataScienceTeam, & Gomilla Salas, J. G. (2023). *Deep Learning de la A a la Z*. Udemy. Recuperado de https://www.udemy.com/course/deep-learning-a-z/?couponCode=ACCAGE0923

[4] Alfaro, M., Calvo, J., Gallego, A. J., Garrido, C., Ríos, A., & Valero, J. J. (2023). *Deep Learning con Keras y Pytorch*. Grupo ANAYA.

[5] Aprende Machine Learning. (2018). Breve historia de las redes neuronales artificiales. Recuperado de https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/#:~:text=1989%20%E2%80%93%20Convolutional%20Neural%20Network&text=La%20primera%20CNN%20fue%20creada,de%20caracter%C3%ADsticas%20y%20luego%20clasificar

[6] Keepcoding. (2024). Arquitectura VGG16 y VGG19 en Deep Learning. Recuperado de https://keepcoding.io/blog/arquitectura-vgg16-vgg19-deep-learning/

[7] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*, 41–48. https://doi.org/10.1145/1553374.1553380

[8] Raduly, Z., Sulyok, C., Vadàszi, Z., & Zölde, A. (2018). Dog breed identification using deep learning. *2018 IEEE International Conference on Future IoT Technologies*, 1-5. https://ieeexplore.ieee.org/document/8524715

[9] Valarmathi, B., Prakash, G., Reddy, R. H., Gupta, N. S., Saravanan, S., & Shanmugasundaram, P. (2023). Hybrid deep learning algorithms for dog breed identification—A comparative analysis. *2023 IEEE International Conference on Data Science and Engineering (ICDSE)*, 1-6. https://ieeexplore.ieee.org/document/10192536

[10] Gunaranjan, K., Vijay, K., & Naveen, P. (2024). Transfer learning and fine-tuned CNN architecture for dog breed classification. *2024 IEEE Conference on Machine Learning and Applications (ICMLA)*, 1-6. https://ieeexplore.ieee.org/document/10574709

[11] Saglietti, L., Mannelli, S. S., & Saxe, A. (2022). An analytical theory of curriculum learning in teacher-student networks. *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 1-15. https://neurips.cc

[12] Sulyok, C. (2018). *Dog Breed Identification Using Deep Learning*. ResearchGate. Recuperado de https://www.researchgate.net/publication/328834665_Dog_Breed_Identification_Using_Deep_Learning

[13] Du, Y. (2021). *Figura 3: The convolutional neural network (CNN) architecture*. ResearchGate. Recuperado de https://www.researchgate.net/figure/The-convolutional-neural-network-CNN-architecture_fig5_350152088

[14] ResearchGate. (2021). *Figura 6: The architecture of the ResNet-50 network*. Recuperado de https://www.researchgate.net/figure/The-architecture-of-the-ResNet-50-network_fig3_356162462

[15] ResearchGate. (2022). *Figura 7: The architecture of the ResNet-101 model*. Recuperado de https://www.researchgate.net/figure/The-architecture-of-the-ResNet-101-model_fig7_360641512

[16] Kramberger, T. (2020). *Figura 8: ResNet-152 neural network architecture*. ResearchGate. Recuperado de https://www.researchgate.net/profile/Tin-Kramberger/publication/343615852/figure/fig1/AS:923861193330692@1597277081980/ResNet-152-neural-network-architecture-Slika-1-Arhitektura-ResNet-152-neuronske-mreze.ppm

[17] ResearchGate. (2021). *Figura 9: Arquitectura VGG16*. Recuperado de https://www.researchgate.net/figure/Figura-10-Arquitectura-VGG16-42_fig5_356119399

Apéndices

Apéndice A: Declaración de la ruta de datos.

```
# Leer los datos de las etiquetas
train_labels = pd.read_csv('C:/Users/jaime/Documents/TFG/dog-breed-identification/labels.csv')

# Rutas de los directorios de entrenamiento y prueba
train_path = 'C:/Users/jaime/Documents/TFG/dog-breed-identification/train'
test_path = 'C:/Users/jaime/Documents/TFG/dog-breed-identification/test'

# Agregar extensión ".jpg" a las IDs de las imágenes
train_labels['id'] = train_labels['id'].apply(lambda x: x + ".jpg")
```

Apéndice B: Función que grafica la cantidad de imágenes de cada raza.

```
# Graficar la distribución de las imágenes por raza
plt.figure(figsize=(12, 6))
breed_counts.plot(kind='bar')
plt.xlabel('Raza de Perro')
plt.ylabel('Número de Imágenes')
plt.title('Distribución de Imágenes de Entrenamiento por Raza')
plt.xticks(rotation=45)
plt.show()
```

Apéndice C: Preprocesamiento de imágenes.

```python
# Función para cargar y preprocesar imágenes
def load_and_preprocess_images(image_paths, target_size=(224, 224)):
    images = []
    for path in image_paths:
        # Cargar la imagen y redimensionarla al tamaño objetivo
        image = load_img(path, target_size=target_size)
        # Convertir la imagen a un array numpy y normalizar los valores de píxeles
        image = img_to_array(image) / 255.0
        images.append(image)
    return np.array(images)
```

Apéndice D: División del conjunto de datos.

```python
# Cargar imágenes de entrenamiento y validación
train_image_paths = [os.path.join(train_path, filename) for filename in train_labels['id']]
validate_image_paths = train_image_paths[:int(len(train_image_paths) * 0.2)]  # Tomar un 20% para validación
train_image_paths = train_image_paths[int(len(train_image_paths) * 0.2):]  # Resto para entrenamiento
train_images = load_and_preprocess_images(train_image_paths)
validate_images = load_and_preprocess_images(validate_image_paths)
```

Apéndice E: Codificación de etiquetas.

```python
# Cargar etiquetas de entrenamiento y validación
train_labels = keras.utils.to_categorical(train_labels['breed'].map({breed: i for i, breed in enumerate(breeds)}))
validate_labels = train_labels[:len(validate_images)]
train_labels = train_labels[len(validate_images):]
```

Apéndice F: Procesado y Entrenamiento del modelo utilizando Curriculum Learning.

```
all_breeds = []  # Lista para acumular las razas de cada fase

for i, breeds in enumerate([fase_breeds_1, fase_breeds_2, fase_breeds_3, fase_breeds_4], start=1):
    # Voy acumulando las razas actuales en all_breeds
    all_breeds.extend(breeds)

    print(f"\nEntrenando en la fase {i} con razas acumuladas: {all_breeds}")

    # Filtro las etiquetas con todas las razas acumuladas hasta la fase actual
    phase_labels = train_labels[train_labels['breed'].isin(all_breeds)]
    phase_image_paths = [os.path.join(train_path, filename) for filename in phase_labels['id']]

    # Cargar y procesar imágenes del conjunto acumulado
    phase_images = load_and_preprocess_images(phase_image_paths)

    # Divido datos en entrenamiento y validación (80% entrenamiento, 20% validación)
    val_split_index = int(len(phase_images) * 0.8)
    train_images, val_images = phase_images[:val_split_index], phase_images[val_split_index:]
    train_labels_encoded = keras.utils.to_categorical(
        phase_labels['breed'].map({breed: i for i, breed in enumerate(all_breeds)}))[:val_split_index]
    val_labels_encoded = keras.utils.to_categorical(
        phase_labels['breed'].map({breed: i for i, breed in enumerate(all_breeds)}))[val_split_index:]

    # Defino el modelo, en este caso MobileNetv2, para ajustar la capa de salida según el número actual de razas
    base_model = tf.keras.applications.MobileNetV2(input_shape=(224, 224, 3), include_top=False, weights='imagenet')
    for layer in base_model.layers:
        layer.trainable = False

    # Aquí se agregan las capas adicionales con el número de clases actual
    x = keras.layers.Flatten()(base_model.output)
    x = keras.layers.Dropout(0.4)(x)
    predictor = keras.layers.Dense(len(all_breeds), activation='softmax')(x)

    # Se crea el modelo completo
    model = tf.keras.models.Model(inputs=base_model.input, outputs=predictor)
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.0004)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

    # Por último, se entrena el modelo en el conjunto acumulado
    history = model.fit(
        train_images, train_labels_encoded,
        validation_data=(val_images, val_labels_encoded),
        epochs=10, batch_size=128
    )
```

Apéndice G: Matriz de confusión del modelo MobileNetV2 utilizando Curriculum learning.