TECNOLÓGICO NACIONAL DE MÉXICO

INTELIGENCIA ARTIFICIAL

Profesor:

Alcaraz Chavez Jesus Eduardo

Nombre:

Kevin Alejandro Cuevas Crisantos

Numero de Control:

21121503

Proyecto 2

Reporte Técnico

Proyecto 2: Clasificación de Imágenes de Animales mediante Redes Neuronales Convolucionales (CNN)

1. Introducción

El presente proyecto tiene como finalidad el desarrollo e implementación de un sistema de clasificación automática de imágenes de animales utilizando Redes Neuronales Convolucionales (CNN). Las CNN constituyen una de las arquitecturas más importantes dentro del campo del aprendizaje profundo, especialmente en tareas de visión por computadora, debido a su capacidad para extraer características espaciales relevantes directamente desde imágenes.

El sistema fue desarrollado en Python, empleando bibliotecas como TensorFlow/Keras, OpenCV, NumPy y Matplotlib, y se estructuró mediante un conjunto de scripts y un notebook principal que permiten construir, entrenar, ajustar y evaluar distintos modelos CNN, tanto desde cero como mediante técnicas de *Transfer Learning*.

2. Objetivo del Proyecto

Desarrollar un pipeline completo de clasificación de imágenes que permita:

- **Clasificar imágenes de animales a partir de un dataset organizado por clases.**
- **Implementar y comparar arquitecturas CNN personalizadas.**
- **Aplicar técnicas de *Data Augmentation* para mejorar la generalización del modelo.**
- **Explorar el uso de *Transfer Learning* con modelos preentrenados.**
- **Evaluar el desempeño del modelo mediante métricas de entrenamiento y validación.**

3. Conjunto de Datos (Dataset)

El dataset utilizado se encuentra estructurado en carpetas, donde cada subcarpeta representa una clase distinta de animal, tales como:

- **Gatos**
- **Perros**
- **Tortugas**
- **Hormigas**
- **Mariquitas**

Esta estructura permite una carga directa de las imágenes y una asignación automática de etiquetas durante el entrenamiento. Para la construcción y limpieza del dataset se desarrollaron scripts auxiliares que permiten:

- **Contar imágenes por clase.**
- **Verificar imágenes corruptas.**
- **Redimensionar imágenes.**
- **Extraer imágenes desde fuentes externas.**

4. Carga y Preprocesamiento de Imágenes

El proyecto implementa múltiples enfoques para la carga y el preprocesamiento de imágenes, lo cual permitió experimentar con distintas configuraciones:

4.1 Lectura de Imágenes

- **Se utilizó cv2.imread() en varios scripts, lo que implica lectura en formato BGR.**
- **En el notebook principal se empleó plt.imread(), que trabaja en RGB.**
- **En algunos casos se realizó conversión explícita de BGR a RGB mediante cv2.cvtColor().**

4.2 Redimensionado

Dependiendo del script y del enfoque del modelo, las imágenes fueron redimensionadas a distintos tamaños, como:

- **64 × 64**

Este proceso permitió analizar el impacto del tamaño de entrada en el rendimiento del modelo.

4.3 Normalización

Se emplearon dos estrategias principales:

- **Normalización clásica dividiendo los valores de los píxeles entre 255.**
- **Uso de preprocess_input() al trabajar con modelos preentrenados como Secuencial.**

4.4 Manejo de Canales

Algunos scripts incluyen validaciones para:

- **Convertir imágenes en escala de grises a RGB.**
- **Manejar imágenes RGBA.**

- **Verificar imágenes inválidas (None) antes de procesarlas.**

5. Arquitecturas de Modelos

El proyecto contempla dos enfoques principales de modelado:

5.1 CNN Implementadas desde Cero

Se desarrollaron modelos CNN personalizados utilizando la API secuencial de Keras, incluyendo:

- **Capas convolucionales (Conv2D)**
- **Funciones de activación (ReLU, LeakyReLU)**
- **Capas de *Pooling***
- **Capas densas totalmente conectadas**

Estos modelos fueron entrenados y almacenados en el archivo de animales.h5.

5.2 Transfer Learning

Se implementó *Transfer Learning* utilizando Secuencial preentrenado con pesos de ImageNet. Este enfoque incluyó:

- **Congelamiento del modelo base.**
- **Uso de GlobalAveragePooling2D.**
- **Adición de capas densas finales adaptadas al número de clases del dataset.**

Este método permitió aprovechar características visuales previamente aprendidas, reduciendo el tiempo de entrenamiento y mejorando el desempeño.

6. Data Augmentation

Para mejorar la capacidad de generalización del modelo, se aplicaron técnicas de *Data Augmentation* mediante ImageDataGenerator, tales como:

- **Rotaciones.**
- **Desplazamientos horizontales y verticales.**
- **Zoom.**
- **Volteo horizontal.**

Estas configuraciones se aplicaron tanto en scripts independientes como directamente en el notebook principal.

7. Entrenamiento y Evaluación

El entrenamiento de los modelos se realizó utilizando:

- **model.fit() con datos preprocesados.**
- **Generadores de datos con *augmentation*.**

Métricas evaluadas:

- **Accuracy**
- **Validation Accuracy**
- **Loss**
- **Validation Loss**

Los resultados se visualizaron mediante gráficas y se complementaron con reportes de clasificación (classification_report) para evaluar el desempeño por clase.

8. Automatización y Scripts Auxiliares

El proyecto incluye diversos scripts que permiten:

- **Automatizar modificaciones en el notebook.**
- **Verificar dependencias.**
- **Ejecutar pruebas rápidas de inferencia.**
- **Restaurar y pulir modelos finales.**

Esta estructura modular facilita la experimentación y el ajuste progresivo del sistema.

9. Problemas Detectados

Durante el desarrollo se identificaron diversos puntos críticos, entre ellos:

- **Inconsistencias en el tamaño de imágenes y normalización.**
- **Mezcla de formatos de color (BGR y RGB).**
- **Falta de un entorno reproducible estandarizado.**
- **Dependencia de rutas fijas (*hardcoded*).**
- **Riesgos al modificar notebooks de forma automática.**

Estos aspectos representan oportunidades claras de mejora y aprendizaje dentro del proyecto.

10. Recomendaciones

Como resultado del análisis, se proponen las siguientes mejoras:

- **Unificar el pipeline de preprocesamiento.**
- **Definir un tamaño estándar de imagen.**
- **Centralizar la lectura y validación de imágenes.**
- **Versionar el mapeo de clases.**
- **Mejorar el manejo de excepciones y logging.**
- **Separar claramente notebooks exploratorios y scripts reproducibles.**

11. Conclusiones

Este proyecto permitió aplicar de manera práctica los fundamentos del aprendizaje profundo y la visión por computadora, abordando desde la construcción del dataset hasta la evaluación de modelos CNN avanzados. La implementación de arquitecturas desde cero y el uso de *Transfer Learning* demostraron la flexibilidad y potencia de las CNN para tareas de clasificación de imágenes.

Además, el proyecto fortaleció habilidades clave como el manejo de datos, la experimentación con modelos, la detección de errores y la mejora continua del pipeline de entrenamiento. En conjunto, este trabajo representa una base sólida para el desarrollo de sistemas inteligentes más robustos y escalables.