Clasificación de Imágenes con Redes Neuronales Convolucionales

Nombre del Estudiante
Universidad Autónoma de Nuevo León
Maestría en Ciencia de Datos
Procesamiento y Clasificación de Datos

Marzo 2025

1 Introducción

Las redes neuronales convolucionales (CNN) se han convertido en una herramienta fundamental para el análisis de imágenes, especialmente en tareas como clasificación, segmentación y detección de objetos.

En esta práctica, desarrollamos una red neuronal convolucional capaz de clasificar imágenes en tres categorías: *perros*, *gatos* y *gallinas*. Esta aplicación permite demostrar la eficacia de las CNN para el reconocimiento de patrones visuales complejos. La mayor parte de las imagenes usadas fueron tomadas para este estudio.

2 Metodología

2.1 Datos

El conjunto de datos utilizado se compone de imágenes distribuidas en tres carpetas distintas: **dogs**, **cats** y **chicken**. Las imágenes se cargaron mediante la biblioteca `OpenCV` y se redimensionaron a un tamaño uniforme de **256x256** pixeles. Se usarón aproximadamente 50 imagenes de cada tipo durante este estudio.

2.2 Procesamiento

Las imágenes se normalizaron dividiendo los valores de los pixeles entre 255. Posteriormente, los datos se dividieron en conjuntos de entrenamiento y prueba en una proporción de 80% y 20%, respectivamente.

2.3 Arquitectura del Modelo

La arquitectura de la red neuronal convolucional (CNN) desarrollada en esta práctica está diseñada para procesar imágenes RGB de 256x256 píxeles y clasificarlas en una de tres categorías: perros, gatos o gallinas. A continuación se describe cada una de las capas y su propósito:

- **Capa de entrada (Input Layer)**: Acepta imágenes con dimensiones $256 \times 256 \times 3$ (ancho, alto, y canales de color RGB). Esta capa simplemente recibe los datos sin hacer procesamiento.
- **Bloques de convolución y pooling (Conv2D + MaxPooling2D)**: Se implementan dos bloques de este tipo.
  - **Conv2D**: Aplica filtros (también llamados kernels) que extraen características locales de la imagen, como bordes, texturas o patrones específicos. Cada filtro genera un mapa de activación que resalta ciertas características visuales.
  - **MaxPooling2D**: Reduce la dimensionalidad espacial de los mapas de activación obtenidos por la convolución, conservando las características más importantes. Esto ayuda a reducir el número de parámetros y el costo computacional, además de aportar cierta invariancia a traslaciones.
- **Capa Flatten**: Convierte los mapas de características tridimensionales obtenidos en una sola dimensión para que puedan ser procesados por las capas densas. Es una transición entre las capas convolucionales y las totalmente conectadas.
- **Capa densa intermedia (Dense)**: Es una capa completamente conectada que permite la combinación de las características aprendidas en las etapas anteriores. Se utiliza una función de activación ReLU para introducir no linealidad.
- **Capa Dropout**: Se incluye para prevenir el sobreajuste. Esta capa desactiva aleatoriamente un porcentaje de neuronas durante el entrenamiento, lo que obliga a la red a aprender representaciones más robustas y a no depender excesivamente de ciertas conexiones.
- **Capa de salida (Output Layer)**: Contiene tres neuronas (una por cada clase), y utiliza la función de activación **softmax**, que convierte los valores de

salida en probabilidades que suman 1. La clase con la mayor probabilidad es tomada como la predicción del modelo.

Esta arquitectura es suficientemente sencilla para evitar sobreajuste en un conjunto de datos pequeño, pero suficientemente compleja para captar las diferencias entre las clases visuales.

3 Resultados

El modelo se entrenó durante varias épocas usando la función de pérdida **categorical crossentropy** y el optimizador **Adam**. Se evaluó el desempeño mediante la métrica de **precisión** en el conjunto de prueba.

A continuación se presenta la evolución de el modelo a lo largo del entrenamiento:

Época	Accuracy	Val_Accuracy	Loss
1	0.3242	0.3667	3.3134
2	0.3467	0.6667	1.1990
3	0.6406	0.7000	0.9532
4	0.7125	0.7667	0.7887
5	0.7512	0.8333	0.6740
6	0.7542	0.6333	0.6210
7	0.7465	0.6000	0.7796
8	0.8587	0.7333	0.4641
9	0.8658	0.7000	0.4147
10	0.9602	0.6667	0.2154

Table 1: Precisión y pérdida a lo largo del entrenamiento.

3.1 Ejemplos de Clasificación

A continuación se presentan ejemplos de predicciones realizadas por el modelo entrenado para cada una de las clases.

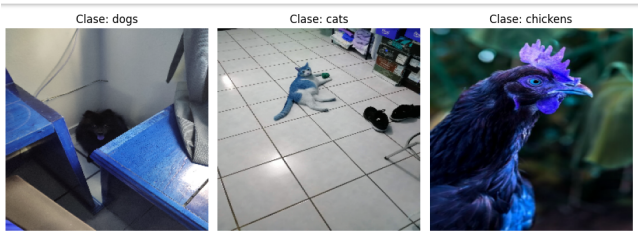

Figure 1: Ejemplos de imágenes clasificadas por el modelo para cada clase: perro, gato y gallina.

4 Conclusiones

La red neuronal convolucional implementada logró clasificar correctamente la mayoría de las imágenes de prueba. Este ejercicio demuestra el potencial de las CNN para tareas de clasificación de imágenes, incluso con un conjunto de datos relativamente sencillo.

Para mejorar los resultados en futuras versiones se podría:

- Aumentar la cantidad de datos con técnicas de *data augmentation*.
- Aplicar regularización adicional para reducir el sobreajuste.
- Probar arquitecturas preentrenadas como VGG o MobileNet.