Tarea 5. Análisis de imágenes para clasificación de animales

Ivan Gabriel Salinas Castillo

19 de febrero de 2025

Resumen

Este documento presenta la implementación y evaluación de una Red Neuronal Convolucional (CNN) entrenada en un subconjunto del dataset CIFAR-10. Se seleccionaron imágenes de tres categorías (gatos, perros y caballos) y se optimizó la arquitectura para su ejecución en Google Colab. Se aplicaron técnicas de aumento de datos y optimización de hiperparámetros, logrando mejorar la precisión del modelo.

1. Introducción

El reconocimiento de imágenes mediante redes neuronales convolucionales (CNN) ha demostrado ser altamente efectivo en tareas de clasificación visual [1]. El dataset CIFAR-10 es un conjunto ampliamente utilizado para benchmarking en visión por computadora [2]. Sin embargo, en entornos con recursos limitados, como Google Colab, es crucial optimizar el uso de datos y parámetros del modelo para maximizar la eficiencia [3].

En este estudio, se entrenó una CNN utilizando únicamente tres clases del conjunto CIFAR-10 (gatos, perros y caballos). Se redujo el tamaño del dataset y se aplicaron técnicas de aumento de datos para mejorar la generalización del modelo. A continuación se muestran los primeros 2 animales de cada categoría en la figura 1.

2. Metodología

2.1. Selección y Preprocesamiento de Datos

Se utilizó el dataset CIFAR-10 [2], extrayendo solo las imágenes correspondientes a las clases de interés. Para reducir el uso de memoria en Google

Colab, se seleccionaron aleatoriamente 2000 imágenes por clase. Posteriormente, las imágenes fueron normalizadas dividiendo sus valores entre 255.

2.2. Arquitectura del Modelo

La arquitectura de la CNN se diseñó considerando la eficiencia computacional. Se implementaron capas de convolución con normalización por lotes (‘BatchNormalization‘), activaciones ‘LeakyReLU‘ y técnicas de regularización (‘Dropout‘). La estructura del modelo fue la siguiente:

- Tres capas convolucionales con filtros de 32, 64 y 128.
- Capas de normalización y activación ‘LeakyReLU‘.
- Capas de agrupación (‘MaxPooling‘).
- Capa completamente conectada con 128 neuronas.
- Capa de salida con activación ‘softmax‘ para clasificación multiclase.

Se utilizó el optimizador Adam con una tasa de aprendizaje de 0.0005 para mejorar la estabilidad del entrenamiento.

2.3. Entrenamiento y Evaluación

El modelo fue entrenado utilizando ‘ImageDataGenerator‘ para aplicar técnicas de aumento de datos (rotaciones, zoom y volteo horizontal). Se emplearon ‘EarlyStopping‘ y ‘ReduceLROnPlateau‘ para evitar el sobre ajuste y mejorar la convergencia.

La evaluación se realizó utilizando la precisión en el conjunto de prueba y la curva ROC para analizar el desempeño de la red en cada categoría.

3. Resultados y Discusión

Las Figuras 2 y 3 muestran la evolución de la precisión y la pérdida del modelo durante el entrenamiento.

Se observa que la precisión en entrenamiento y validación mejoró significativamente en las primeras 10 épocas. Sin embargo, a partir de la época 15, el modelo mostró una tendencia a estabilizarse, indicando una posible saturación del aprendizaje.

Para evaluar la discriminación del modelo, se generó la curva ROC para cada clase como se muestra en la figura 4.

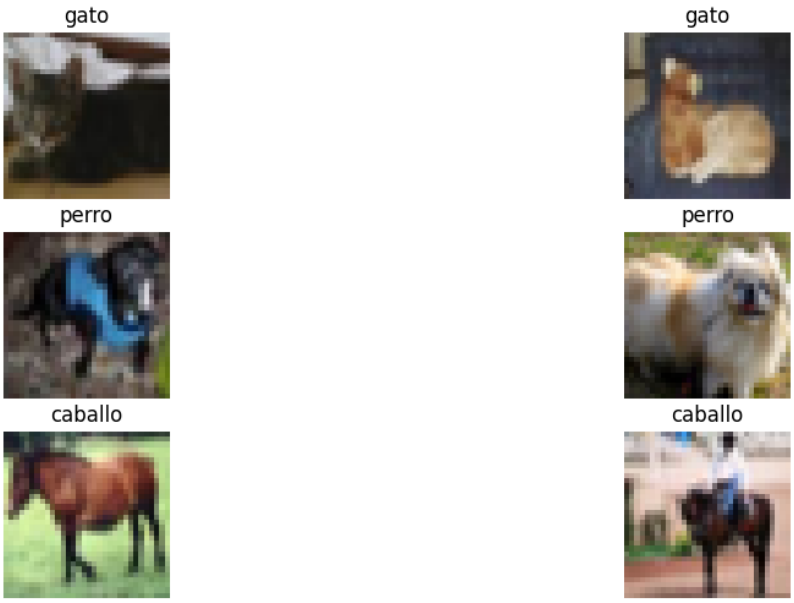

Figura 1: Muestra de datos.

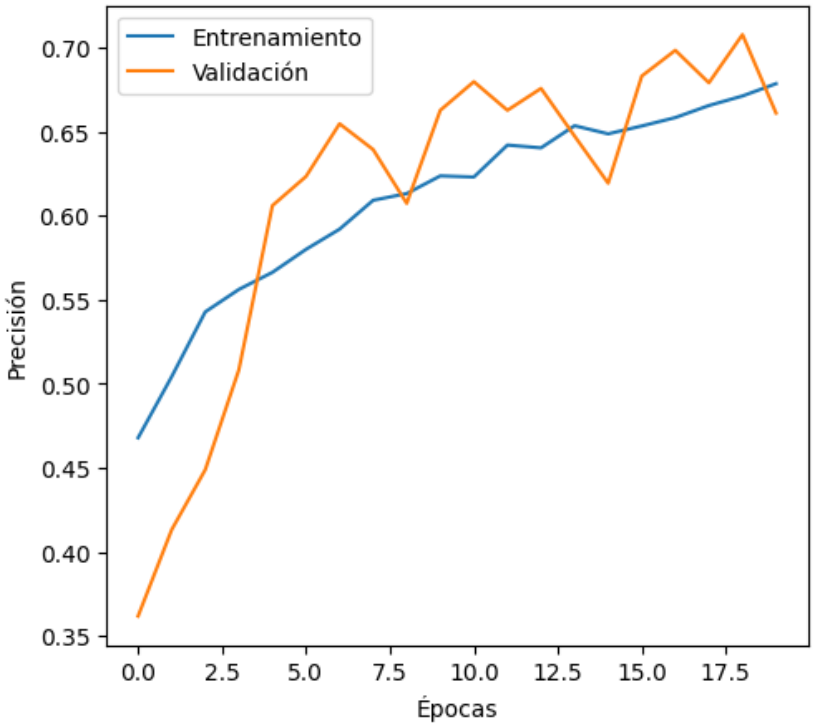

Figura 2: Curva de precisión durante el entrenamiento.

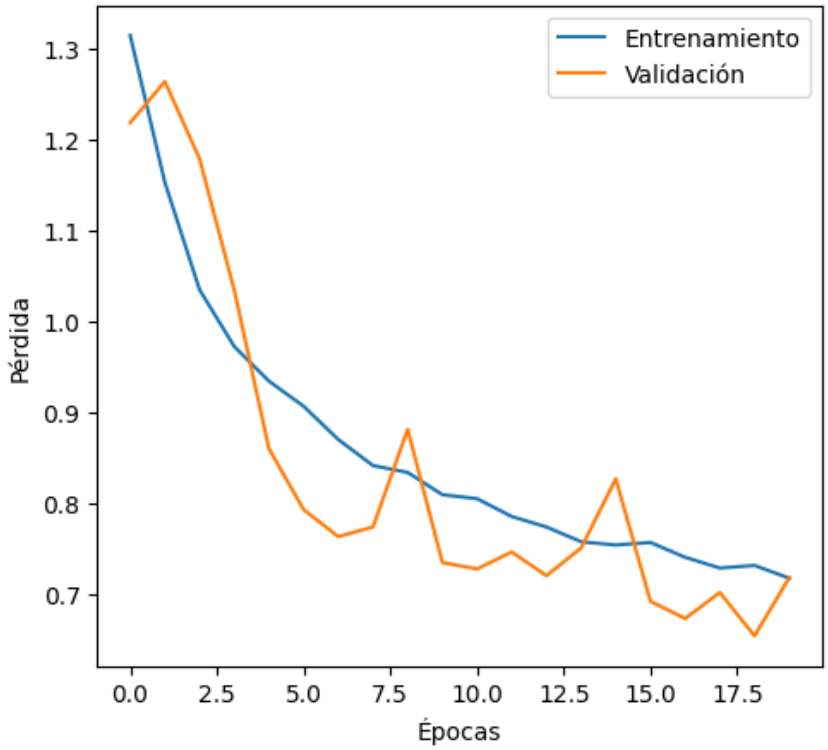

Figura 3: Curva de perdida durante el entrenamiento.

Las áreas bajo la curva (AUC) fueron superiores a 0.80 en todas las clases, indicando un buen desempeño del modelo. La clase caballo obtuvo la mejor discriminación, seguida de gato y perro. A continuación se muestran algunas imágenes con sus valores reales y predichos en la figura 5.

4. Conclusiones

Este estudio demostró que es posible entrenar una CNN eficiente en Google Colab con una cantidad reducida de imágenes. La implementación de técnicas como 'BatchNormalization', 'LeakyReLU' y 'Dropout' mejoró la estabilidad del entrenamiento. La aplicación de 'ImageDataGenerator' permitió aumentar la variabilidad del conjunto de datos y mejorar la generalización.

El análisis de curvas ROC confirmó un buen desempeño del modelo. Futuras mejoras podrían incluir la optimización de hiperparámetros mediante 'GridSearchCV' o el uso de arquitecturas preentrenadas como 'MobileNetV2' para mejorar la eficiencia computacional.

Referencias

[1] Alex Krizhevsky, Ilya Sutskever y Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". En: *Advances in Neural Information Processing Systems* 25 (2012).

[2] Alex Krizhevsky. *The CIFAR-10 dataset*. Available at `https://www.cs.toronto.edu/~kriz/cifar.html`. 2009.

[3] François Chollet. *Deep Learning with Python*. Manning Publications, 2017.

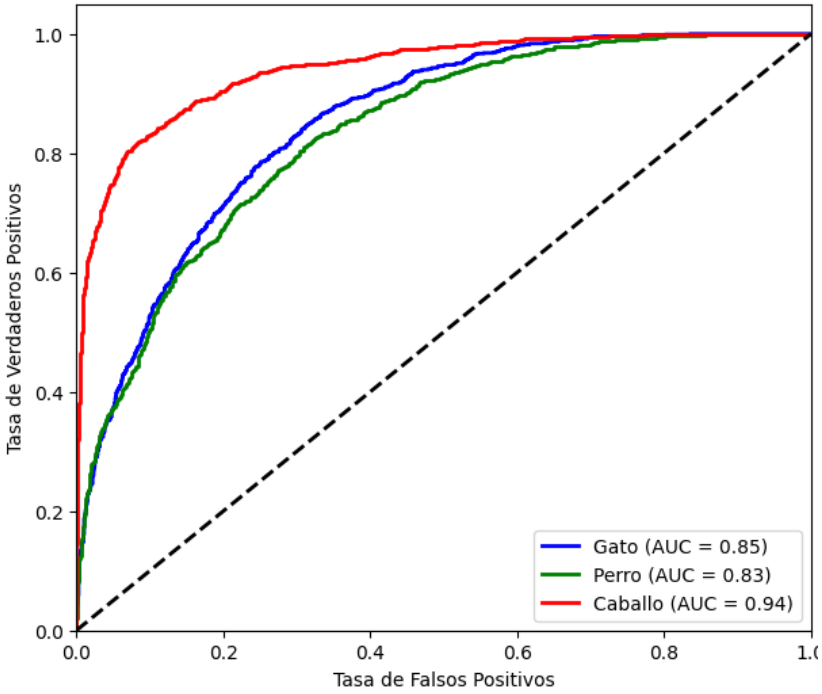

Figura 4: Curvas ROC por clase.

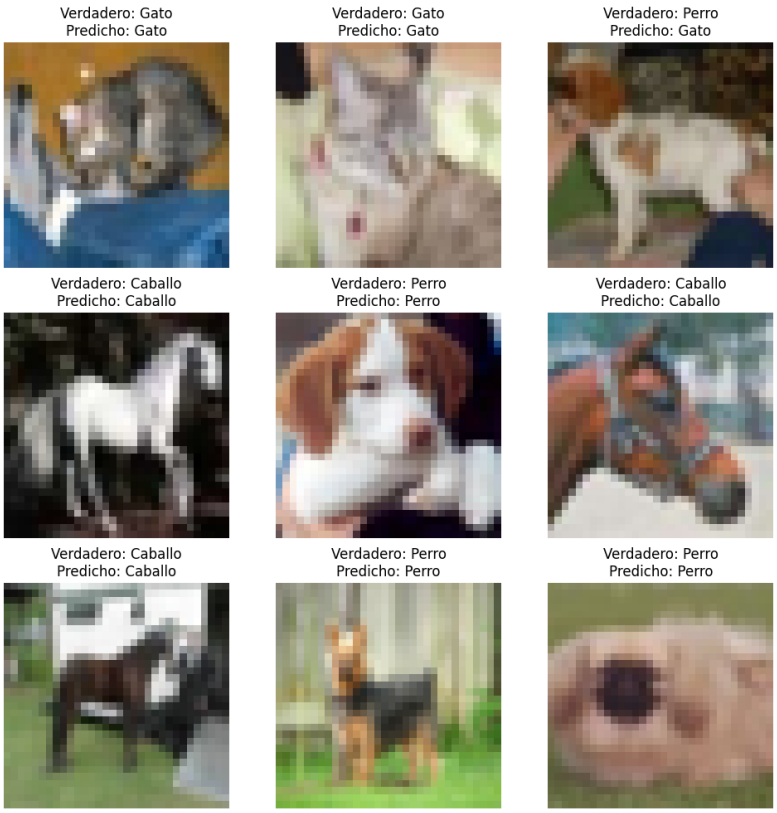

Figura 5: Resultados de la clasificación.