**Sistema de Clasificación de Gatos y Perros con CNN**

Autores: Amalia Roscío Villena Romaní / Leonardo Daniel Ramirez Galindo

Curso: Talleres - Universidad Peruana de Ciencias Aplicadas

Docente: Carlos Fernando Montoya Cubas

Fecha: 20 de abril de 2025

Link github: [**https://github.com/leoraga11/Talleres-Trabajo-final/upload**](https://github.com/leoraga11/Talleres-Trabajo-final/upload)

Link Deploy: https://www.youtube.com/watch?v=Yq18gMI12VE

**Resumen Ejecutivo**

Este proyecto toca temas de visión por computadora y aprendizaje profundo, como redes neuronales. En ese sentido, el objetivo de este proyecto es la clasificación binaria de imágenes: distinguir entre gatos y perros. Para este caso, se ha utilizado la potencia de las Redes Neuronales Convolucionales (CNN). De hecho, se ha construido un sistema capaz de analizar patrones visuales complejos para discernir entre estas dos especies. Así, los hallazgos obtenidos revelan un modelo con una precisión bastante buena, abriendo la puerta a diversas aplicaciones prácticas y señalando caminos futuros para la sofisticación y la interpretabilidad en casos similares.

**1. Introducción y Definición del Problema**

En los temas de la inteligencia artificial, la capacidad de las máquinas para "ver" y comprender el mundo visual que nos rodea ha experimentado una revolución en las últimas décadas. Dentro de esta tarea, la capacidad de diferenciar un gato de un perro significa desafíos fundamentales en el reconocimiento de patrones, la extracción de características relevantes y la generalización a nuevos casos.

Este proyecto aborda precisamente este desafío: construir un sistema que, dada una imagen, pueda determinar con certeza si representa a un gato o a un perro. La relevancia de esta tarea está principalmente relacionada a la curiosidad académica.

La evolución de las técnicas para abordar este tipo de problemas ha sido fascinante. Inicialmente, se dependía de la extracción manual de características (como la forma de las orejas, la textura del pelaje, etc.) combinada con clasificadores tradicionales. Sin embargo, el advenimiento de las CNN marcó un punto de inflexión, permitiendo a las máquinas aprender jerarquías de características directamente de los datos brutos de los píxeles, superando las limitaciones de los métodos manuales y abriendo nuevas posibilidades en la complejidad de las tareas visuales que las máquinas podían abordar.

**1.2. Justificación del Problema y el Dataset "Perros vs. Gatos" de Kaggle**

La selección del problema de clasificación binaria entre gatos y perros no fue aleatoria. De hecho, se fundamenta en los siguientes puntos:

* El dataset "Dogs vs. Cats" representa un conjunto de datos para automatización de CNN. Además, tiene documentación detallada que permite una fácil replicabilidad y comparación con otros estudios.
* El dataset se encontraba bastante balanceado lo que se traduce a que contenía un número aproximadamente igual de imágenes para cada clase. Esto es crucial para el entrenamiento de un modelo imparcial. De no ser el caso, se necesitaría aplicar técnicas de balanceo como SMOTE.
* Finalmente, este problema sirve como una excelente opción de enseñanza dentro del curso, pues permite aplicar conceptos fundamentales de las CNN y experimentar el ciclo de vida completo de un proyecto de aprendizaje automático, desde la preparación de los datos hasta la evaluación y el despliegue.

**2. Descripción y Preprocesamiento del Dataset**

* El dataset, accesible a través de la plataforma Kaggle, contiene un conjunto de 25,000 imágenes etiquetadas, divididas equitativamente entre 12,500 fotografías de gatos y 12,500 de perros. Además, contiene una buena variabilidad dado que abarca diversas razas, poses, condiciones de iluminación y fondos.
* Para construir un modelo confiable y evaluar su capacidad de generalización, dividimos el dataset original en tres subconjuntos distintos:
  + Conjunto de Entrenamiento (70%): Con 17,500 imágenes, la red neuronal ajusta sus pesos internos para mapear las características visuales a las categorías de "gato" o "perro".
  + Conjunto de Validación (15%): Este conjunto, compuesto por 3,750 imágenes nunca vistas por el modelo durante el entrenamiento, es crucial para evaluar el rendimiento y así identificar cuándo el modelo comienza a memorizar los datos de entrenamiento en lugar de aprender patrones generalizables.
  + Conjunto de Prueba (15%): Finalmente, el conjunto de prueba, también con 3,750 imágenes, se separa para la evaluación final del modelo entrenado.
* Antes de alimentar las imágenes a nuestra red neuronal, aplicamos una serie de transformaciones esenciales:
  + Redimensionamiento a 150x150 píxeles: Esta estandarización asegura que todas las imágenes de entrada tengan la misma dimensión.
  + Normalización de Píxeles (Rescaling 1./255): Al dividir los valores de los píxeles (que originalmente varían entre 0 y 255) por 255, los escalamos al rango de 0 a 1.
  + Aumento de Datos (Data Augmentation): Para enriquecer nuestro conjunto de entrenamiento y hacer que el modelo sea más robusto ante diversas variaciones en las imágenes reales, aplicamos técnicas de aumento de datos como rotaciones sutiles, zooms ligeros y volteos horizontales.

**3. Arquitectura del Aprendizaje: Construcción y Entrenamiento del Modelo CNN**

* Arquitectura básica
  + Entrada 150 × 150 px, 3 canales (RGB).
  + Bloques Conv2D (32 y 64 filtros 3×3, ReLU) → MaxPooling 2×2.
  + Flatten → Dense 512 ReLU → Dropout 0.5 → Salida sigmoide (1 nodo).
* Hiperparámetros y entrenamiento
  + Optimizador Adam, lr 0.001; pérdida binary cross‑entropy.
  + Batch 32; hasta 15 épocas con *early stopping* (paciencia 3).
* Mejoras y control de sobreajuste
  + Aumento de datos: rotaciones suaves, zoom y volteo horizontal.
  + ReduceLROnPlateau para bajar la tasa de aprendizaje si la validación se estanca.

**4. Evaluación del Rendimiento**

Tras el entrenamiento, medimos qué tan bien generaliza el modelo con imágenes nunca vistas. Para ello usamos métricas clásicas de clasificación binaria y una inspección rápida de errores comunes.

* Accuracy: porcentaje de aciertos globales.
* Precision: de las imágenes predichas como “perro”, cuántas lo son de verdad.
* Recall: de todos los “perros” reales, cuántos detectamos.
* F1‑score: equilibrio entre precision y recall.
* Matriz de confusión: TP, TN, FP, FN para ver dónde falla.

|  |  |  |
| --- | --- | --- |
| Métrica | Entrenamiento | Validación |
| Accuracy | 90.76 % | 73.41 % |
| Precision | 91.67 % | 71.35 % |
| Recall | 89.38 % | 78.26 % |

Los números indican cierto sobre‑ajuste: el modelo rinde ~91 % en entrenamiento, pero baja a ~73 % en validación.

* **Errores típicos**
  + Gatos con orejas puntiagudas → FP (gato → perro).
  + Perros pequeños o cachorros → FN (perro → gato).

**5. Despliegue del Modelo**

* **Interfaz con Streamlit**

Creamos una app web ligera donde cualquiera puede probar el clasificador sin instalar nada más que un navegador.

* **Flujo de uso**
  1. El usuario sube una imagen (JPG/PNG).
  2. La app redimensiona y normaliza la foto.
  3. La CNN genera la probabilidad “Gato” / “Perro”.
  4. Mostramos la etiqueta y la confianza (%) en pantalla.
* **Ejecución y hosting**
  1. Local: streamlit run app.py abre la demo en http://localhost:8501.
  2. En línea: el mismo script puede subirse sin cambios a Streamlit Cloud, Heroku o Render para compartir el modelo con otros usuarios.

**Conclusión**

Este proyecto confirmó que una CNN básica puede resolver la clasificación gato‑perro con un desempeño aceptable en imágenes nuevas. Aun así, la brecha entre entrenamiento y validación señala sobreajuste y deja claro que hay espacio para optimizar.

**Líneas de mejora prioritarias**

* **Aumento de datos avanzado**: aplicar transformaciones más variadas (brillos, recortes, oclusiones) para hacer al modelo menos sensible a cambios de escena.
* **Transfer learning**: ajustar redes pre‑entrenadas (VGG 16, ResNet, EfficientNet) para ganar precisión con menos esfuerzo de entrenamiento.
* **Nuevas arquitecturas**: evaluar Visual Transformers y modelos híbridos CNN‑Transformer que ya rinden bien en visión.
* **Despliegue escalable**: contenerizar la app con Docker y migrarla a un servicio cloud para facilitar su uso y actualización.

En conclusión, el trabajo demuestra la utilidad práctica de las CNN y marca un punto de partida sólido; las acciones listadas permitirán llevar el sistema a mayor precisión, transparencia y alcance real.