Ejercicio: Clasificación de Imágenes de Perros y Gatos usando una CNN en Python

Objetivo

Desarrollar una red neuronal convolucional (CNN) utilizando Python y TensorFlow/Keras para clasificar imágenes de perros y gatos empleando un dataset público.

Competencias a desarrollar

- Uso de Deep Learning para clasificación de imágenes.
- Manejo de datasets públicos.
- Preprocesamiento de imágenes.
- Construcción y entrenamiento de CNNs.
- Evaluación de modelos de Machine Learning.
- Predicción sobre nuevas imágenes.

Requisitos Previos

Software requerido

- Python 3.10+
- Jupyter Notebook o Google Colab
- Librerías:
    - tensorflow
    - keras
    - matplotlib
    - numpy

Instalación

Opción 1: Google Colab

No requiere instalación.

Opción 2: Instalación local

```
pip install tensorflow matplotlib numpy
```

Dataset Público

Utilizaremos el dataset público:

Dogs vs Cats Dataset

Contiene miles de imágenes etiquetadas de perros y gatos.

Dataset oficial:
Dogs vs Cats Dataset (Kaggle)

Estructura esperada

Después de descargar y descomprimir:

```
dogs-vs-cats/
│
├── train/
│   ├── cats/
│   └── dogs/
│
└── test/
    ├── cats/
    └── dogs/
```

Fundamento Teórico

¿Qué es una CNN?

Una CNN (Convolutional Neural Network) es un tipo de red neuronal especializada en procesamiento de imágenes.

Las CNN utilizan:

- Capas convolucionales
- Funciones de activación
- Pooling
- Capas densas

para aprender patrones visuales automáticamente.

Desarrollo de la práctica

Paso 1. Importar bibliotecas

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
```

Paso 2. Cargar dataset

Definir rutas

```
train_dir = "dogs-vs-cats/train"
test_dir = "dogs-vs-cats/test"
```

Cargar imágenes

```
train_dataset = tf.keras.utils.image_dataset_from_directory(
    train_dir,
    image_size=(150,150),
    batch_size=32
)

test_dataset = tf.keras.utils.image_dataset_from_directory(
    test_dir,
    image_size=(150,150),
    batch_size=32
)
```

Paso 3. Visualizar imágenes

```
class_names = train_dataset.class_names
print(class_names)
plt.figure(figsize=(10,10))

for images, labels in train_dataset.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Paso 4. Normalización

```
normalization_layer = layers.Rescaling(1./255)

train_dataset = train_dataset.map(
    lambda x, y: (normalization_layer(x), y)
)

test_dataset = test_dataset.map(
    lambda x, y: (normalization_layer(x), y)
)
```

Paso 5. Construcción de la CNN

```
model = models.Sequential()

# Capa convolucional 1
model.add(layers.Conv2D(32, (3,3), activation='relu',
                        input_shape=(150,150,3)))
model.add(layers.MaxPooling2D((2,2)))

# Capa convolucional 2
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))

# Capa convolucional 3
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))

# Aplanamiento
model.add(layers.Flatten())

# Capas densas
model.add(layers.Dense(128, activation='relu'))

# Salida
model.add(layers.Dense(1, activation='sigmoid'))
```

Paso 6. Resumen del modelo

```
model.summary()
```

Paso 7. Compilación

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

Paso 8. Entrenamiento

```
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=test_dataset
)
```

Paso 9. Evaluación del modelo

```
test_loss, test_acc = model.evaluate(test_dataset)

print("Precisión:", test_acc)
```

Paso 10. Graficar resultados

Precisión

```
plt.plot(history.history['accuracy'], label='Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Validación')

plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Pérdida

```
plt.plot(history.history['loss'], label='Entrenamiento')
plt.plot(history.history['val_loss'], label='Validación')

plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Paso 11. Predicción con nuevas imágenes

```
from tensorflow.keras.preprocessing import image
```

Cargar imagen

```
img = image.load_img(
    "mi_imagen.jpg",
    target_size=(150,150)
)

img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = img_array / 255.0
```

Predicción

```
prediction = model.predict(img_array)

if prediction[0][0] > 0.5:
    print("Perro")
else:
    print("Gato")
```

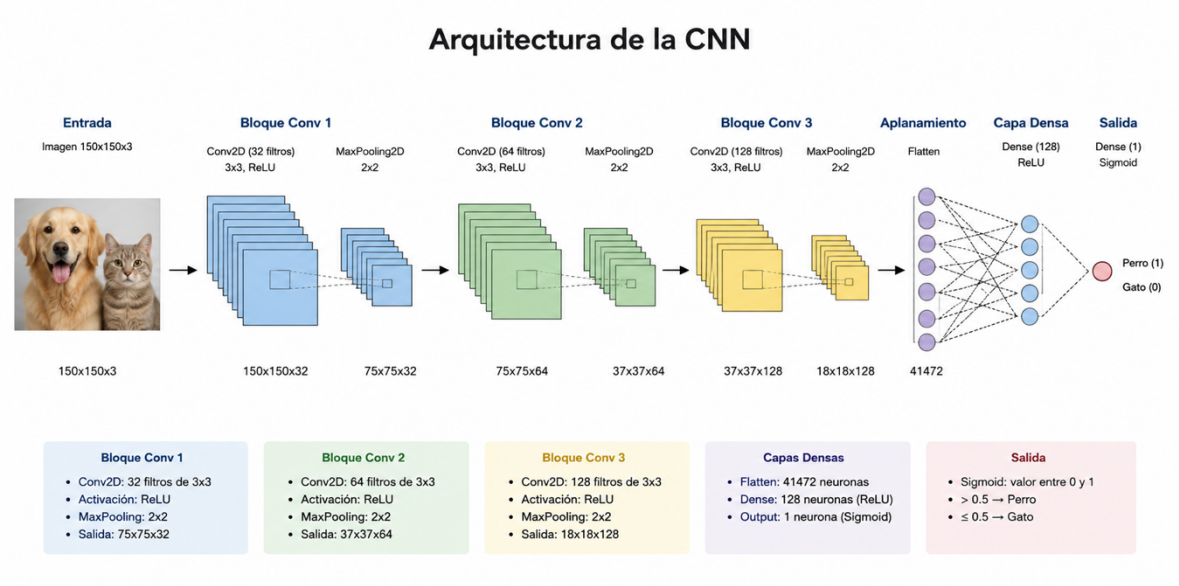
Explicación de Arquitectura

La CNN utilizada contiene:

Capa	Función
Conv2D	Detecta patrones
MaxPooling	Reduce dimensiones
Flatten	Convierte a vector
Dense	Clasificación
Sigmoid	Salida binaria

Arquitectura

A continuación se muestra la arquitectura del clasificador de imágenes:



Resultados Esperados

El modelo debe alcanzar aproximadamente:

- 80% – 90% de precisión

dependiendo de:

- número de épocas
- capacidad del equipo
- tamaño del dataset

Actividades Complementarias

Actividad 1

Cambiar:

- tamaño de imágenes
- número de filtros
- épocas

y comparar resultados.

Actividad 2

Agregar más capas convolucionales.

Actividad 3

Implementar:

- Dropout
- Data Augmentation

para reducir overfitting.

Ejemplo con Data Augmentation

```
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])
```

Preguntas de reflexión

1. ¿Qué función tienen las capas convolucionales?
2. ¿Qué ocurre si aumentamos las épocas?
3. ¿Qué es overfitting?
4. ¿Por qué se normalizan las imágenes?
5. ¿Qué ventaja tiene una CNN sobre una red neuronal tradicional?

Reto Extra

Modificar el proyecto para clasificar:

- caballos y vacas
- frutas
- flores
- vehículos

usando datasets públicos.

Conclusión

En esta práctica se desarrolló una CNN capaz de clasificar imágenes de perros y gatos utilizando TensorFlow/Keras y un dataset público. Además, se aplicaron conceptos fundamentales de visión computacional y Deep Learning.

Recursos adicionales

TensorFlow

TensorFlow Oficial

Keras

Keras Documentation

Dataset Dogs vs Cats

Dogs vs Cats Dataset (Kaggle)