EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE SAN LUIS POTOSÍ

SISTEMAS COMPUTACIONALES

Evaluación

NOMBRE DEL DOCENTE:

CORDERO MARTINEZ STEPHANIE

NOMBRE DEL ALUMNO:

ALVARADO GUTIERREZ EMILIO DEL ANGEL

Materia: INTELIGENCIA ARTIFICIAL

Hora: 5:00 – 6:00

Fecha: 10/12/2024

Contenido

Reporte: Clasificación de Especies de Animales (Perros, Gatos, Aves).

Objetivo:

El objetivo de este proyecto es desarrollar un sistema de clasificación de imágenes de especies animales mediante redes neuronales entrenadas con TensorFlow y OpenCV. Se ha implementado un modelo para predecir a qué especie pertenece una imagen de perro, gato o pájaro.

Alcances del Proyecto

Alcance 1: Carga de imágenes y predicción utilizando TensorFlow

Este código se encarga de entrenar un modelo de clasificación de imágenes utilizando la red neuronal VGG16, preentrenada en el conjunto de datos de ImageNet, y luego realiza una predicción sobre una imagen cargada desde el disco.

1. **Carga y Preprocesamiento de Datos**:
    - Se utiliza ImageDataGenerator para cargar y preprocesar las imágenes. Las imágenes de entrenamiento se cargan desde un directorio especificado (ruta de entrenamiento) y se dividen en un 80% para entrenamiento y 20% para validación.
2. **Configuración del Modelo**:
    - Se añaden capas adicionales (Flatten para aplanar los datos y Dense para las capas totalmente conectadas) para adaptar el modelo a nuestro conjunto de clases (en este caso, tres clases: Perro, Gato y Ave).
3. **Entrenamiento del Modelo**:
    - El modelo es compilado utilizando el optimizador Adam y la función de pérdida categorical_crossentropy, adecuada para problemas de clasificación multiclase.
    - Se entrena el modelo con los datos de entrenamiento, utilizando la validación en el conjunto de validación para evaluar su rendimiento.
4. **Evaluación del Modelo**:
    - Una vez entrenado, el modelo es evaluado sobre un conjunto de prueba independiente para medir su precisión.

5. **Predicción**:
    - Se carga una imagen desde el directorio especificado y se preprocesa de manera similar a las imágenes de entrenamiento. Luego, se realiza la predicción utilizando el modelo entrenado.
    - La predicción es convertida a una clase, utilizando un diccionario de etiquetas que mapea las clases numéricas a sus nombres correspondientes: 0 -> Perro, 1 -> Gato, 2 -> Ave.
6. **Evidencia:**
    - Código:

```
import numpy as np
import tensorflow as tf
import cv2  # OpenCV para cargar imágenes
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications import VGG16
from tensorflow.keras.optimizers import Adam

# Configuración para cargar y normalizar las imágenes, con separación para validación
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)  # 80% entrenamiento, 20% validación

train_generator = train_datagen.flow_from_directory(
    r'C:\Users\USUARIO DELL\Desktop\Inteligencia Artificial\Evaluacion\dataset\train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    r'C:\Users\USUARIO DELL\Desktop\Inteligencia Artificial\Evaluacion\dataset\train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)

```

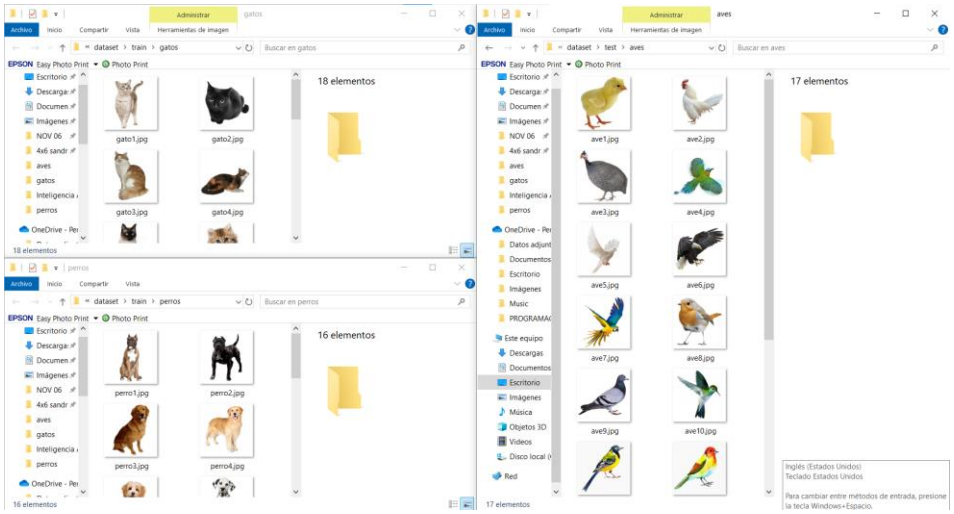
- Imagen a predecir:

```
35.# Ruta a la imagen para predecir (ajusta la ruta según la imagen que
   quieras predecir)
36.img_path = 'C:/Users/USUARIO DELL/Desktop/Inteligencia
   Artificial/Evaluacion/dataset/train/gatos/gato1.jpg'
```

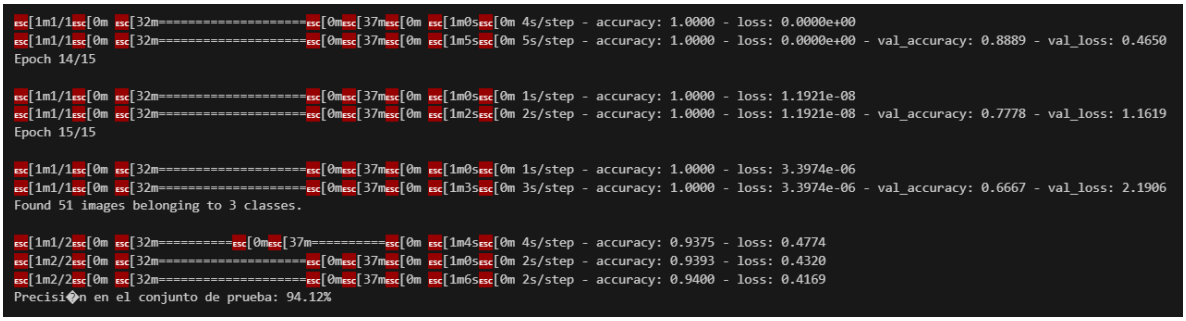
La imagen será la de un gato.

- Dataset:

```
dataset/
    ├── train/
    │   ├── aves/
    │   │   ├── ave1.jpg
    │   │   ├── ave2.jpg
    │   │   ├── ave3.jpg
    │   ├── gatos/
    │   │   ├── gato1.jpg
    │   │   ├── gato2.jpg
    │   │   ├── gato3.jpg
    │   ├── perros/
    │   │   ├── perro1.jpg
    │   │   ├── perro2.jpg
    │   │   ├── perro3.jpg
    ├── test/
    │   ├── aves/
    │   │   ├── ave1_test.jpg
    │   │   ├── ave2_test.jpg
    │   ├── gatos/
    │   │   ├── gato1_test.jpg
    │   │   ├── gato2_test.jpg
    │   ├── perros/
    │   │   ├── perro1_test.jpg
    │   │   ├── perro2_test.jpg
```

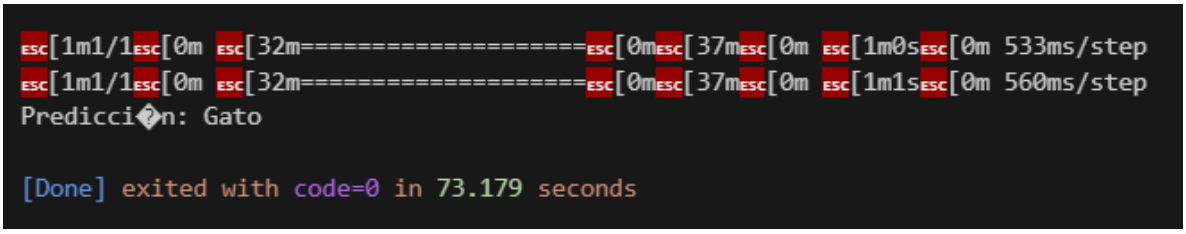


- Output código épocas:



Con un total de 51 imágenes y tres clases: Aves, Gatos y Perros; Y una precisión del 94.12%.

- Predicción



Predicción: Gato.

El sistema permite cargar imágenes y realizar predicciones sobre ellas utilizando un modelo preentrenado de redes neuronales. Las imágenes fueron cargadas desde el código y el modelo fue entrenado para clasificar las imágenes en una de las tres categorías: perro, gato o ave.

Alcance 2: Predicción utilizando una cámara web

El código que se muestra está dividido en varias secciones que permiten la carga y entrenamiento de un modelo de clasificación de imágenes, la predicción de imágenes utilizando ese modelo y la integración con una cámara web para capturar imágenes en tiempo real.

1. Carga y entrenamiento del modelo

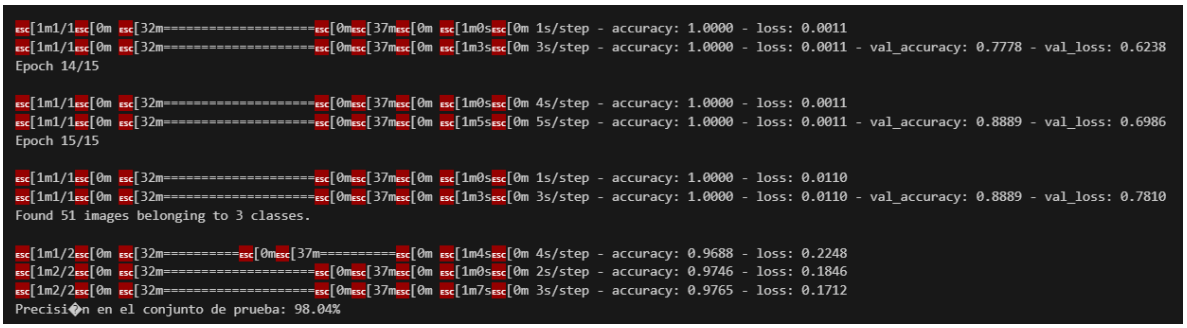
- **Cargar y preprocesar las imágenes**:
    - Utiliza la clase ImageDataGenerator de Keras para cargar las imágenes desde las carpetas correspondientes (entrenamiento y validación), y realiza la normalización (escalar los valores de píxel a un rango de 0 a 1).
- **Entrenamiento del modelo**:
    - El modelo es entrenado durante 15 épocas con el optimizador Adam y la función de pérdida categorical_crossentropy, adecuada para clasificación multiclase.
    - El modelo utiliza los generadores de entrenamiento y validación para aprender las características de las imágenes y realizar la clasificación.
- **Captura de imágenes en tiempo real**:
    - Se utiliza cv2.VideoCapture(0) para acceder a la cámara web y capturar fotogramas en tiempo real.
    - Los fotogramas se muestran en una ventana de OpenCV usando cv2.imshow().
- **Toma de fotos y predicción**:
    - Cuando el usuario presiona la tecla "s" en el teclado, el programa guarda la imagen actual de la cámara en un archivo local llamado "captured_image.png".
    - La imagen preprocesada se pasa al modelo para hacer la predicción. El modelo devuelve las probabilidades de cada clase, y la clase con la mayor probabilidad se selecciona como la predicción.
    - La predicción se muestra en la imagen capturada y se imprime en la consola.
- **Salir del bucle**:
    - Si se presiona la tecla "q", el programa termina el bucle de captura y cierra las ventanas de OpenCV.

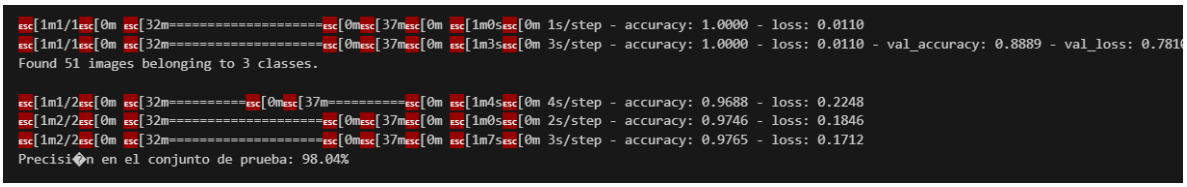
- **Evidencia:**
    - Codigo:

```
# Inicia la cámara web
cap = cv2.VideoCapture(0)

while True:
    # Captura el fotograma de la cámara
    ret, frame = cap.read()
    if not ret:
        break
```

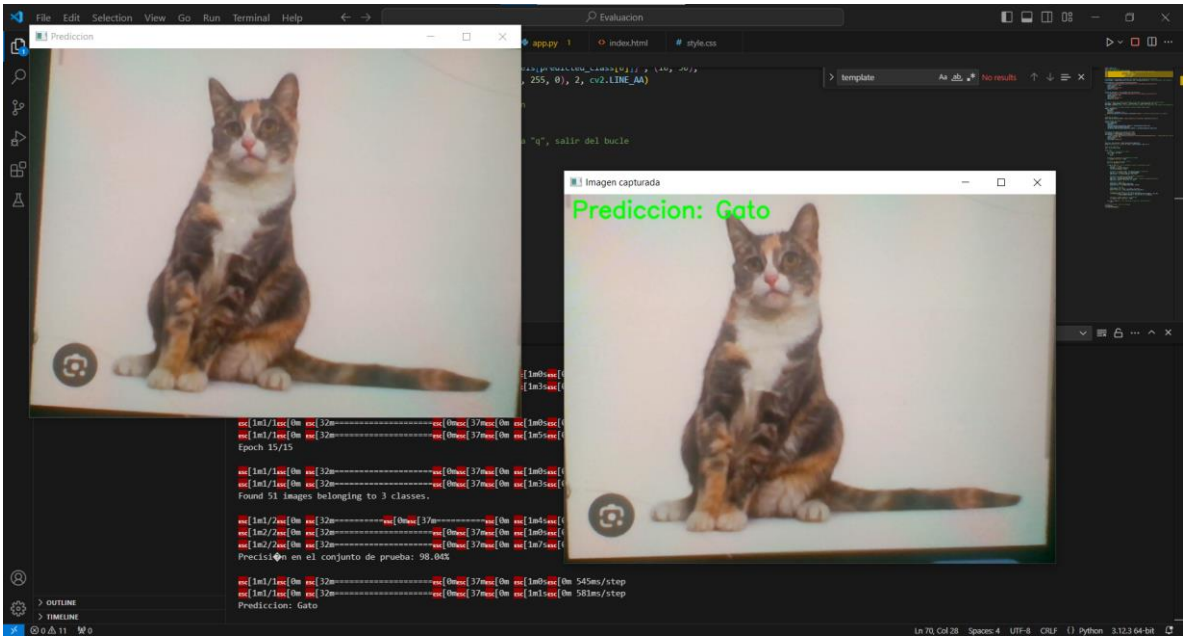
    - OutPud código épocas:



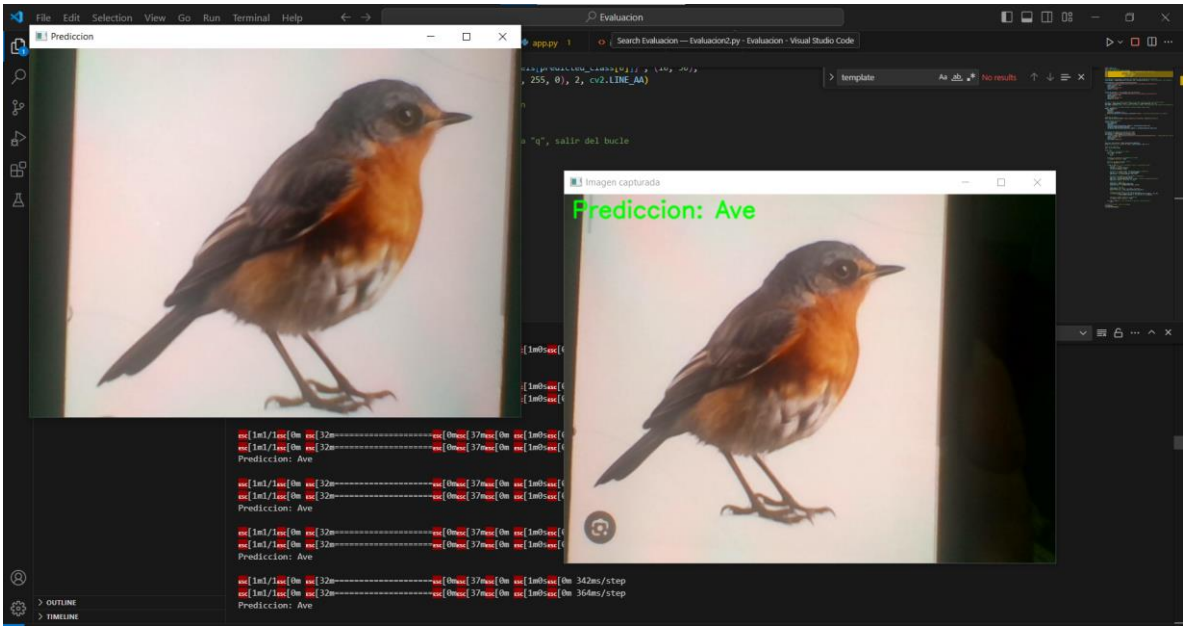


Se usa la letra “s” para tomar captura y la letra “q” para salir del programa

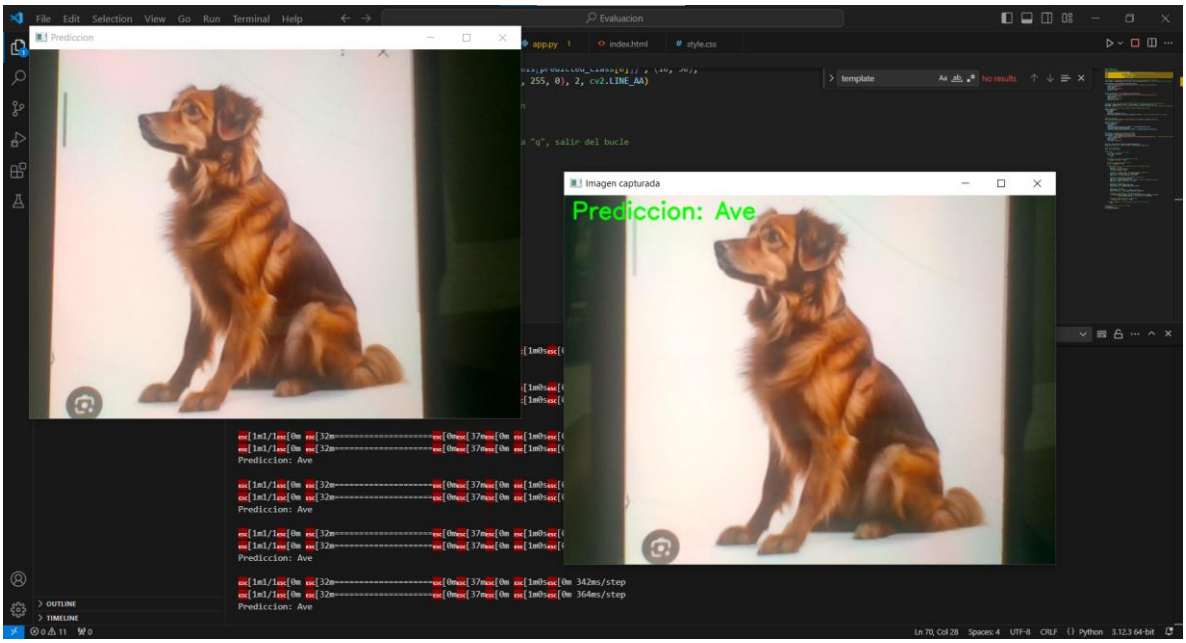
- Capturas de predicciones



Predicción de gato.



Predicción de ave.



Predicción de perro (fallida).