INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Computación Gráfica, visión computacional y multimedia*				
TÍTULO DE LA PRÁCTICA:	*Clasificación y Reconocimiento*				
NÚMERO DE PRÁCTICA:	*11*	**AÑO LECTIVO:**	*2023-A*	**NRO. SEMESTRE:**	*IX*
FECHA DE PRESENTACIÓN	*16/07/2023*	**HORA DE PRESENTACIÓN**	*08:30 pm*		
INTEGRANTE (s) *Luis Diego Valdivia Turpo*				**NOTA (0-20)**	
DOCENTE(s): *Diego Alonso Iquira Becerra*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS PROPUESTOS:

1. Descripción general del programa

El programa implementa un sistema de análisis de emociones faciales en tiempo real utilizando la cámara de un dispositivo. Utiliza técnicas de procesamiento de imágenes y aprendizaje profundo para detectar rostros en los cuadros de video capturados y clasificar las emociones asociadas a cada rostro utilizando un modelo pre-entrenado. A medida que se captura el video, se dibujan rectángulos alrededor de los rostros detectados y se superponen etiquetas con las emociones predichas. Este enfoque permite identificar y visualizar las emociones en tiempo real, lo que puede ser útil en aplicaciones de análisis emocional y reconocimiento facial. Al presionar la tecla 'q', se sale del programa y se liberan los recursos de la cámara y las ventanas abiertas. En resumen, el programa proporciona una forma interactiva y automatizada de detectar y analizar las emociones faciales en tiempo real a partir de imágenes capturadas por la cámara de un dispositivo.

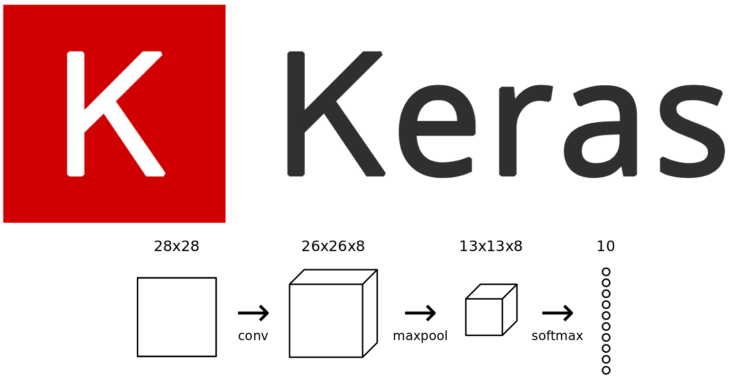
2. Tecnologías utilizadas

a. *OpenCV (Open Source Computer Vision Library)*

OpenCV es una biblioteca de código abierto ampliamente utilizada para el procesamiento de imágenes y la visión por computadora. Proporciona herramientas y funciones para trabajar con imágenes y videos, realizar detección y seguimiento de objetos, reconocimiento de patrones, entre otros. En este código, OpenCV se utiliza para capturar video en tiempo real desde la cámara, realizar la detección de rostros utilizando el clasificador de cascada Haar y dibujar rectángulos delimitadores en los rostros detectados.

b. *Keras*

Keras es una biblioteca de alto nivel escrita en Python que facilita la creación y el entrenamiento de redes neuronales. Proporciona una interfaz sencilla y modular para construir modelos de aprendizaje profundo y se ejecuta sobre el backend de TensorFlow. En este código, Keras se utiliza para cargar y utilizar un modelo pre-entrenado de red neuronal convolucional (CNN) para la clasificación de emociones faciales.



c. NumPy (Numerical Python)

NumPy es una biblioteca fundamental para la computación científica en Python. Proporciona soporte para matrices y operaciones matemáticas de alta velocidad. En este código, NumPy se utiliza para realizar operaciones numéricas en matrices, como expandir dimensiones y realizar cálculos de índices.

d. Matplotlib

Matplotlib es una biblioteca de trazado y visualización de datos en Python. Proporciona una amplia variedad de funciones y herramientas para crear gráficos, histogramas, diagramas de dispersión, entre otros. En este código, Matplotlib se utiliza para mostrar y visualizar las imágenes capturadas por la cámara junto con los rectángulos delimitadores y las etiquetas de las emociones predichas.

e. Deep learning

Deep learning es una rama del campo de la inteligencia artificial (IA) que se centra en el desarrollo y entrenamiento de modelos de redes neuronales artificiales profundas. Estos modelos se componen de múltiples capas y conexiones neuronales, lo que les permite aprender y extraer características complejas de conjuntos de datos a gran escala.

3. Ventajas y desventajas

a. *Ventajas*

- Análisis en tiempo real
- Automatización del proceso, ya que el programa utiliza técnicas de detección de rostros y modelos pre-entrenados para clasificar las emociones faciales automáticamente
- El análisis de emociones faciales en tiempo real puede tener aplicaciones en una amplia gama de industrias. Por ejemplo, en el campo de la investigación psicológica, el programa puede utilizarse para estudiar las respuestas emocionales de las personas en diferentes situaciones. En la industria del entretenimiento, puede aplicarse para mejorar la interacción y la experiencia del usuario en videojuegos y aplicaciones interactivas. Además, en el ámbito de la publicidad y el marketing, puede utilizarse para comprender las reacciones emocionales de los consumidores frente a productos o campañas.

b. *Desventajas*

- Precisión del modelo, ya que esta depende en gran medida de la calidad del modelo pre-entrenado utilizado. Si el modelo no ha sido entrenado adecuadamente o no es lo suficientemente preciso, es posible que las emociones predichas no sean precisas
- El rendimiento del programa puede verse afectado por condiciones de iluminación deficientes o baja calidad de video. Si la iluminación es demasiado brillante o demasiado tenue, o si la calidad del video es baja, la detección de rostros y la precisión del análisis de emociones pueden verse comprometidas.
- Limitaciones de la detección de rostros: El programa depende de la detección de rostros utilizando el clasificador de cascada Haar. Si el clasificador no puede detectar correctamente los rostros en ciertas situaciones, como ángulos de cámara difíciles, oclusión parcial del rostro, o diferentes posiciones y expresiones faciales, el programa puede no funcionar de manera óptima.

4. Funciones

a. *Importación de bibliotecas:*

En esta sección se importan todas las bibliotecas necesarias para el programa, incluyendo OpenCV, Keras, NumPy y Matplotlib. Estas bibliotecas proporcionan las funcionalidades

requeridas para el procesamiento de imágenes, la detección de rostros, el aprendizaje profundo y la visualización de resultados.

```
import os
import cv2
import numpy as np
from keras.preprocessing import image
import warnings
warnings.filterwarnings("ignore")
from keras.preprocessing.image import load_img, img_to_array
from keras.models import  load_model
import matplotlib.pyplot as plt
import numpy as np
```

b. Carga del modelo pre-entrenado:

En esta sección se carga el modelo pre-entrenado utilizando la función load_model() de Keras. El modelo se carga desde un archivo llamado "best_model.h5" y se almacena en la variable model. Este modelo se utilizará más adelante para predecir las emociones faciales.

```
model = load_model("best_model.h5")
```

c. Carga del clasificador de cascada Haar para detección de rostros:

Aquí se carga el clasificador de cascada Haar, que se utilizará para detectar rostros en las imágenes capturadas por la cámara. El clasificador se carga utilizando la clase CascadeClassifier de OpenCV y se asigna a la variable face_haar_cascade.

```
face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
```

d. Inicialización de la captura de video:

Esta sección inicializa la captura de video utilizando la cámara del dispositivo. Se crea un objeto VideoCapture de OpenCV con el parámetro 0 para indicar que se utilizará la cámara predeterminada. Además, se crea un bucle while True para capturar continuamente los cuadros de video de la cámara.

```
cap = cv2.VideoCapture(0)
```

e. Procesamiento de la imagen y detección de rostros:

Dentro del bucle, se lee cada cuadro de video utilizando la función cap.read(). El cuadro de video se almacena en la variable test_img. A continuación, se convierte la imagen de BGR a RGB utilizando cv2.cvtColor(). Luego, se utiliza el clasificador de cascada Haar para detectar los rostros en la imagen mediante la función detectMultiScale(), que devuelve las coordenadas de los rostros detectados.

```
while True:
    ret, test_img = cap.read()
    if not ret:
        continue
    gray_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)

    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32, 5)
```

f. Preprocesamiento de la región de interés (ROI) y predicción de emociones:

Para cada rostro detectado, se realiza el siguiente procesamiento:

- Se extrae la región de interés (ROI) que contiene el rostro de la imagen original.
- La ROI se redimensiona a un tamaño específico utilizando cv2.resize().
- Se realiza la conversión de la ROI a una matriz de píxeles utilizando image.img_to_array() de Keras.
- La matriz de píxeles se preprocesa normalizando y expandiéndose en una dimensión adicional.
- Finalmente, se utiliza el modelo pre-entrenado para predecir la emoción en la ROI utilizando model.predict().

```
for (x, y, w, h) in faces_detected:
    cv2.rectangle(test_img, (x, y), (x + w, y + h), (255, 0, 0), thickness=7)
    roi_gray = gray_img[y:y + w, x:x + h]
    roi_gray = cv2.resize(roi_gray, (224, 224))
    img_pixels = image.img_to_array(roi_gray)
    img_pixels = np.expand_dims(img_pixels, axis=0)
    img_pixels /= 255

    predictions = model.predict(img_pixels)

    max_index = np.argmax(predictions[0])

    emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
    predicted_emotion = emotions[max_index]
```

g. Visualización de resultados:

Para cada rostro detectado, se dibuja un rectángulo alrededor del rostro utilizando cv2.rectangle(). Además, se superpone la etiqueta de la emoción predicha en el rectángulo utilizando cv2.putText(). Luego, se muestra la imagen resultante con los rectángulos y etiquetas utilizando cv2.imshow(). También se redimensiona la imagen para ajustarla a una ventana de visualización específica.

```
    cv2.putText(test_img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

resized_img = cv2.resize(test_img, (1000, 700))
cv2.imshow('Facial emotion analysis ', resized_img)
```

h. Salida del bucle y liberación de recursos:

El bucle se ejecuta hasta que se presiona la tecla 'q'. En ese momento, se sale del bucle y se liberan los recursos utilizando cap.release() y cv2.destroyAllWindows(). Esto cierra la cámara y las ventanas de visualización.

```
    if cv2.waitKey(10) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows
```

5. Enlace con el video de demostración

En el siguiente enlace se puede visualizar el video correspondiente a este laboratorio:
https://drive.google.com/file/d/16gqe9s-ymYrerhHDBWtussUfwDpn2AX5/view?usp=sharing

6. Enlace con los scripts del laboratorio

En el siguiente enlace se pueden descargar los scripts que se usaron para este laboratorio:
https://drive.google.com/drive/folders/1zn9CwetSbCu3Z6VuBuiPoIOadursUscz?usp=sharing

II. CUESTIONARIO:

- ***Describir un algoritmo utilizado para el reconocimiento y clasificación de objetos***
  - ➢ ***Algoritmo de clasificación basado en aprendizaje profundo utilizando redes neuronales convolucionales (CNN).***
    - *Recopilación de datos: Se recopilan imágenes etiquetadas de los objetos que se desean reconocer y clasificar. Estas imágenes se dividen en un conjunto de entrenamiento y un conjunto de prueba.*
    - *Preprocesamiento de datos: Las imágenes se redimensionan a un tamaño específico y se normalizan los valores de píxeles para asegurar que estén en un rango adecuado para el modelo.*
    - *Entrenamiento del modelo:*
      a. *Se construye una CNN, que consta de múltiples capas convolucionales, capas de agrupación y capas completamente conectadas.*
      b. *Se inicializan los pesos de la red neuronal.*
      c. *Se realiza un proceso iterativo de entrenamiento en el conjunto de entrenamiento. En cada iteración (época):*
         - *Se pasa un lote de imágenes a través de la red neuronal.*
         - *Se calcula la salida de la red y se compara con las etiquetas verdaderas.*
         - *Se ajustan los pesos de la red utilizando un algoritmo de optimización, como descenso de gradiente estocástico (SGD) o Adam, para minimizar la función de pérdida.*
      d. *Se repite el proceso de entrenamiento hasta que se alcance un criterio de convergencia, como un número máximo de épocas o una mejora mínima en la función de pérdida.*
    - *Evaluación del modelo: Se evalúa el rendimiento del modelo en el conjunto de prueba. Las imágenes de prueba se pasan a través de la red neuronal y se comparan las predicciones con las etiquetas verdaderas. Se calculan métricas de*

evaluación, como precisión, recall y exactitud, para evaluar la capacidad del modelo para reconocer y clasificar objetos correctamente.

- *Uso del modelo: Una vez entrenado y evaluado, el modelo se puede utilizar para reconocer y clasificar objetos en nuevos datos. Las imágenes de entrada se pasan a través de la red neuronal y se obtienen las predicciones de clase correspondientes.*

- ***¿Qué problemas pueden ocurrir al realizar el reconocimiento y clasificación de objetos?***
  - *Variabilidad en los objetos: Los objetos pueden tener variaciones en apariencia, tamaño, posición, iluminación y orientación, lo que dificulta su reconocimiento preciso. Esto puede llevar a errores en la clasificación cuando los objetos presentan diferencias significativas con respecto a los datos de entrenamiento.*
  - *Falta de datos de entrenamiento: La disponibilidad de datos de entrenamiento adecuados y representativos es esencial para entrenar modelos precisos. Si hay una falta de datos o los datos de entrenamiento son sesgados o insuficientes, el modelo puede tener dificultades para generalizar y clasificar correctamente objetos nuevos.*
  - *Overfitting: El overfitting ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento específicos y no puede generalizar bien a datos nuevos. Esto puede resultar en un rendimiento deficiente en la clasificación de objetos no vistos previamente.*
  - *Dificultad en la selección de características: La selección de características adecuadas y representativas puede ser un desafío en el reconocimiento de objetos. Si las características elegidas no son lo suficientemente descriptivas, el modelo puede tener dificultades para reconocer y clasificar correctamente los objetos.*
  - *Clasificación ambigua: Algunos objetos pueden presentar características o apariencias similares, lo que puede llevar a una clasificación ambigua. Esto puede ser especialmente problemático cuando se trata de objetos que comparten características visuales cercanas o pertenecen a clases similares.*
  - *Costo computacional: Dependiendo del tamaño del conjunto de datos y la complejidad del modelo utilizado, el reconocimiento y clasificación de objetos pueden requerir un alto costo computacional. Esto puede limitar la velocidad de procesamiento o la capacidad de implementación en entornos con recursos limitados.*

CONCLUSIONES

- Se debe destacar la importancia del reconocimiento de objetos en diversas áreas, como la visión por computadora, la robótica, la seguridad y el procesamiento de imágenes. En cuanto a las aplicaciones prácticas de esta tecnología, como la detección de emociones faciales en tiempo real, estas aplicaciones

demuestran cómo el reconocimiento de objetos puede ser utilizado para comprender y responder a las características y comportamientos de los objetos en el mundo real.

- Existen desafíos y problemas asociados con el reconocimiento y clasificación de objetos, como la variabilidad en los objetos, la falta de datos de entrenamiento, el overfitting y la clasificación ambigua. Estos desafíos resaltan la complejidad y la importancia de abordar factores como la calidad de los datos, la selección de características adecuadas y el ajuste adecuado de los modelos para obtener resultados precisos y confiables.
- Fue importante reconocer el uso de redes neuronales convolucionales (CNN) como modelo para el reconocimiento y clasificación de objetos. El aprendizaje profundo ha demostrado un gran potencial en el campo del reconocimiento de objetos, permitiendo la extracción de características complejas y la mejora del rendimiento en diversas tareas de visión por computadora. Este enfoque ha demostrado ser efectivo en el análisis de emociones faciales y en otras aplicaciones que requieren un procesamiento avanzado de datos.

METODOLOGÍA DE TRABAJO

REFERENCIAS Y BIBLIOGRAFÍA

[1] Gordon, V. S., & Clevenger, J. L. (2020). Computer Graphics Programming in OpenGL with C++. Stylus Publishing, LLC.

[2] D. P. Kothari, G. Awari, D. Shrimankar , A. Bhende (2019), Mathematics for Computer Graphics and Game Programming: A Self-Teaching Introduction