Azure Computer Vision API

Es un servicio de Microsoft Azure que permite analizar imágenes y detectar objetos, personas, y más. Requiere una clave de suscripción y un endpoint.

Ventajas:

Precisión alta.
Soporte para múltiples características como descripción de imágenes, detección de objetos, OCR, etc.

Desventajas:

No es completamente gratuito (tiene un nivel gratuito limitado).

Requiere conexión a internet y configuración de una cuenta de Azure.

Hugging Face Transformers (Modelo facebook/detr-resnet-50)

Es un modelo de detección de objetos basado en DETR (DEtection TRansformer) que se ejecuta localmente utilizando PyTorch. No requiere conexión a internet ni claves de API.

Ventajas:

Completamente gratuito.
Se ejecuta localmente, lo que garantiza privacidad.
Fácil de usar con la biblioteca transformers de Hugging Face.

Desventajas:

Requiere instalar dependencias adicionales como torch y transformers.

Puede ser más lento en máquinas sin GPU.

Clarifai

Clarifai es una plataforma de inteligencia artificial especializada en visión por computadora y procesamiento de imágenes/videos, que ofrece APIs y herramientas para el reconocimiento de objetos, clasificación de contenido y más.

Ventajas

Fácil integración

Ofrece APIs bien documentadas y SDKs para múltiples lenguajes (Python, JavaScript, etc.), lo que facilita su implementación en aplicaciones.

Compatibilidad con plataformas como AWS, Google Cloud y Azure.

Modelos preentrenados

Proporciona modelos listos para usar en detección de objetos, reconocimiento facial, moderación de contenido (NSFW), texto en imágenes, etc.

Útil para proyectos rápidos sin necesidad de entrenar modelos desde cero.

Personalización

Permite entrenar modelos personalizados con tus propios datos, adaptándose a necesidades específicas.

Desventajas

Costo elevado en escalamiento

Aunque tiene un plan gratuito (limitado a 1,000 operaciones/mes), los precios suben rápidamente según el volumen de consultas.

Puede ser caro para startups o proyectos pequeños.

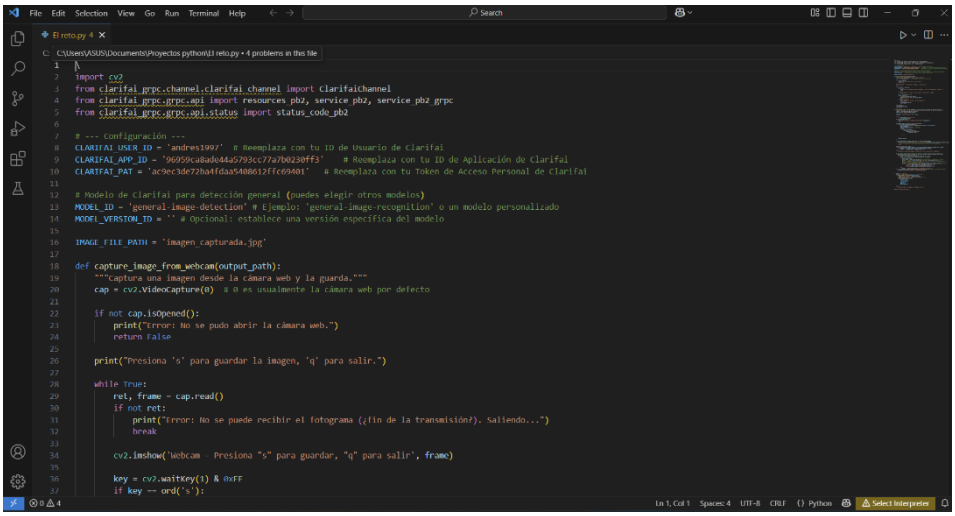
Precisión limitada en casos complejos

Sus modelos preentrenados no siempre son tan precisos como soluciones custom (ej. detección de objetos en ángulos raros o imágenes con ruido).

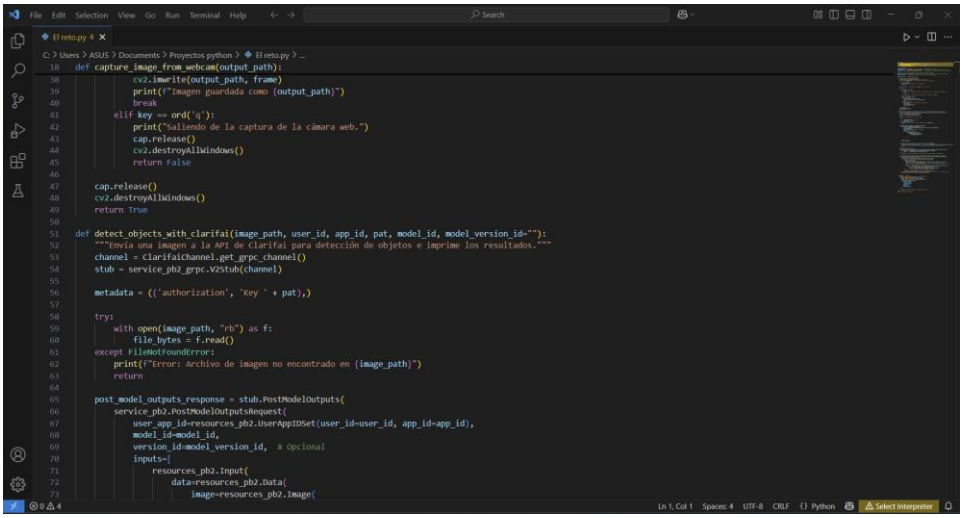
Dependencia de la nube

La mayoría de sus servicios son cloud-based, lo que puede ser un problema si necesitas procesamiento offline.

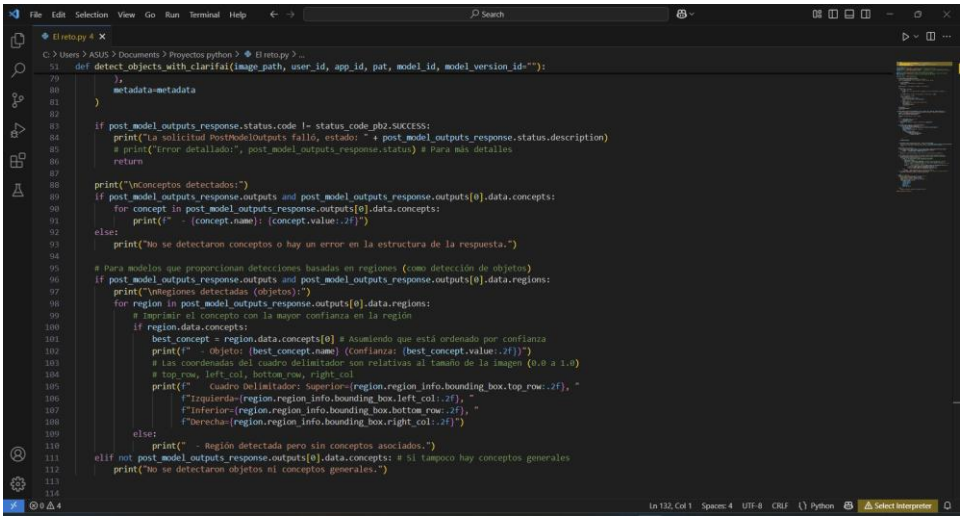
Imágenes del código y funcionamiento



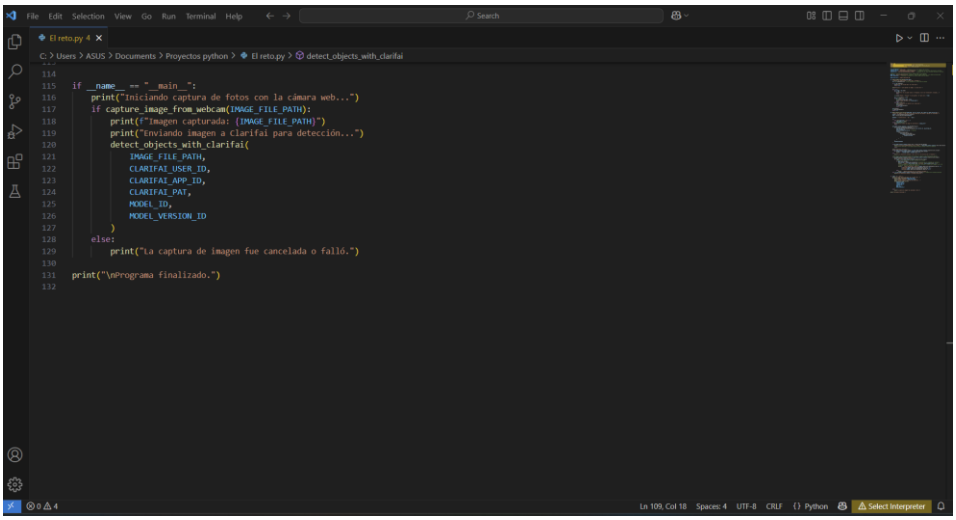
```python
import cv2
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import resources_pb2, service_pb2_grpc
from clarifai_grpc.grpc.api.status import status_code_pb2

# --- Configuración ---
CLARIFAI_USER_ID = 'andres1997'  # Reemplaza con tu ID de Usuario de Clarifai
CLARIFAI_APP_ID = '96059ca8ade44a5793cc77a7b0230ff3'   # Reemplaza con tu ID de Aplicación de Clarifai
CLARIFAI_PAT = 'ac9ec3de72ba4fdaa5408612ffc69401'  # Reemplaza con tu Token de Acceso Personal de Clarifai

# Modelo de Clarifai para detección general (puedes elegir otros modelos)
MODEL_ID = 'general-image-detection' # Ejemplo: 'general-image-recognition' o un modelo personalizado
MODEL_VERSION_ID = '' # Opcional: establece una versión específica del modelo

IMAGE_FILE_PATH = 'imagen_capturada.jpg'

def capture_image_from_webcam(output_path):
    """Captura una imagen desde la cámara web y la guarda."""
    cap = cv2.VideoCapture(0)  # 0 es usualmente la cámara web por defecto

    if not cap.isOpened():
        print("Error: No se pudo abrir la cámara web.")
        return False

    print("Presiona 's' para guardar la imagen, 'q' para salir.")

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Error: No se puede recibir el fotograma (¿fin de la transmisión?). Saliendo...")
            break

        cv2.imshow('Webcam - Presiona "s" para guardar, "q" para salir', frame)

        key = cv2.waitKey(1) & 0xFF
        if key == ord('s'):
```



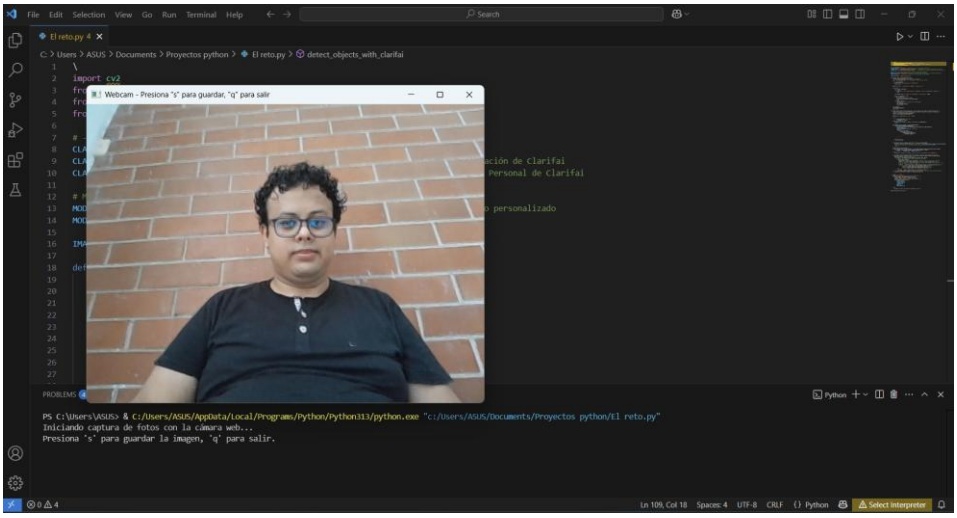
```python
def capture_image_from_webcam(output_path):
            cv2.imwrite(output_path, frame)
            print(f"Imagen guardada como {output_path}")
            break
        elif key == ord('q'):
            print("Saliendo de la captura de la cámara web.")
            cap.release()
            cv2.destroyAllWindows()
            return False

    cap.release()
    cv2.destroyAllWindows()
    return True

def detect_objects_with_clarifai(image_path, user_id, app_id, pat, model_id, model_version_id=""):
    """Envía una imagen a la API de Clarifai para detección de objetos e imprime los resultados."""
    channel = ClarifaiChannel.get_grpc_channel()
    stub = service_pb2_grpc.V2Stub(channel)

    metadata = (('authorization', 'Key ' + pat),)

    try:
        with open(image_path, "rb") as f:
            file_bytes = f.read()
    except FileNotFoundError:
        print(f"Error: Archivo de imagen no encontrado en {image_path}")
        return

    post_model_outputs_response = stub.PostModelOutputs(
        service_pb2.PostModelOutputsRequest(
            user_app_id=resources_pb2.UserAppIDSet(user_id=user_id, app_id=app_id),
            model_id=model_id,
            version_id=model_version_id,  # Opcional
            inputs=[
                resources_pb2.Input(
                    data=resources_pb2.Data(
                        image=resources_pb2.Image(
```



```python
def detect_objects_with_clarifai(image_path, user_id, app_id, pat, model_id, model_version_id=""):
            ),
            metadata=metadata
        )

        if post_model_outputs_response.status.code != status_code_pb2.SUCCESS:
            print("La solicitud PostModelOutputs falló, estado: " + post_model_outputs_response.status.description)
            # print("Error detallado:", post_model_outputs_response.status) # Para más detalles
            return

        print("\nConceptos detectados:")
        if post_model_outputs_response.outputs and post_model_outputs_response.outputs[0].data.concepts:
            for concept in post_model_outputs_response.outputs[0].data.concepts:
                print(f"  - {concept.name}: {concept.value:.2f}")
        else:
            print("No se detectaron conceptos o hay un error en la estructura de la respuesta.")

        # Para modelos que proporcionan detecciones basadas en regiones (como detección de objetos)
        if post_model_outputs_response.outputs and post_model_outputs_response.outputs[0].data.regions:
            print("\nRegiones detectadas (objetos):")
            for region in post_model_outputs_response.outputs[0].data.regions:
                # Imprimir el concepto con la mayor confianza en la región
                if region.data.concepts:
                    best_concept = region.data.concepts[0] # Asumiendo que está ordenado por confianza
                    print(f"  - Objeto: {best_concept.name} (Confianza: {best_concept.value:.2f})")
                    # Las coordenadas del cuadro delimitador son relativas al tamaño de la imagen (0.0 a 1.0)
                    # top_row, left_col, bottom_row, right_col
                    print(f"    Cuadro Delimitador: Superior={region.region_info.bounding_box.top_row:.2f}, "
                          f"Izquierda={region.region_info.bounding_box.left_col:.2f}, "
                          f"Inferior={region.region_info.bounding_box.bottom_row:.2f}, "
                          f"Derecha={region.region_info.bounding_box.right_col:.2f}")
                else:
                    print("  - Región detectada pero sin conceptos asociados.")
        elif not post_model_outputs_response.outputs[0].data.concepts: # Si tampoco hay conceptos generales
            print("No se detectaron objetos ni conceptos generales.")
```



```
if __name__ == "__main__":
    print("Iniciando captura de fotos con la cámara web...")
    if capture_image_from_webcam(IMAGE_FILE_PATH):
        print(f"Imagen capturada: {IMAGE_FILE_PATH}")
        print("Enviando imagen a Clarifai para detección...")
        detect_objects_with_clarifai(
            IMAGE_FILE_PATH,
            CLARIFAI_USER_ID,
            CLARIFAI_APP_ID,
            CLARIFAI_PAT,
            MODEL_ID,
            MODEL_VERSION_ID
        )
    else:
        print("La captura de imagen fue cancelada o falló.")

print("\nPrograma finalizado.")
```



```
import cv2
```

Webcam - Presiona "s" para guardar, "q" para salir

```
PS C:\Users\ASUS> & C:/Users/ASUS/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/ASUS/Documents/Proyectos python/El reto.py"
Iniciando captura de fotos con la cámara web...
Presiona 's' para guardar la imagen, 'q' para salir.
```