



PROGRAMACION

Integrantes:

Diego Andres Escobar Aguilar Carnet:0905-24-21379

Edgar Emanuel Valenzuela Pinto Carnet:0905-24-16186

Laboratorio Reto

Para este Trabajo en clase se nos asignó el reto de desarrollar un programa en Python capaz de tomar una imagen desde la cámara web y analizarla mediante una API de inteligencia artificial, con el objetivo de determinar si hay una persona presente en la imagen. Como parte del proceso, debíamos investigar diferentes APIs y seleccionar la más adecuada para nuestro caso.

Investigación






Mucha de la información fue hecha por mi compañero Edgar.

Investigación de APIs Exploramos varias APIs enfocadas en reconocimiento de imágenes. Usamos ChatGPT como asistente para obtener una comparativa inicial y recomendaciones. Este fue el prompt utilizado:

prompt:

mira tengo este laboratorio quiero que me digas que me recomiendas primero a la hora de elegir la api para hacer el laboratorio y porque
Requisitos del Programa:
u Captura de Imagen: Desarrollar un programa que tome una foto a través de la cámara web de la computadora.
Uso de API de IA: Investigar y seleccionar una API de inteligencia artificial que permita subir la imagen capturada y determinar si hay una persona en la foto.
u Resultados Esperados: El programa debe mostrar el resultado de la detección (si hay una persona o no).
u Opcional: Ampliar el programa para que describa otros objetos identificados en la imagen.
Investigación: Deben investigar diferentes plataformas que ofrecen APIs gratuitas o de prueba para la detección de personas y objetos. Algunas opciones incluyen:
u Clarifai u Roboflow u Google Cloud Vision API u Azure Computer Vision u Hugging Face.

Esto fue la respuesta de chat GPT:

Prioridad	API	¿Por qué?
	Roboflow	Tiene modelos pre entrenados, detección de personas, REST API, plan gratuito.
	Hugging Face	Muy fácil de usar, puedes probar modelos de detección de objetos directamente.
	Clarifai	Buena para detección general, pero requiere más configuración.
	Google Vision	Potente pero más complejo de iniciar, requiere cuenta con facturación.
	Azure Vision	Similar a Google, buena API pero más pasos para autenticación.

Para obtener una segunda opinión le pregunté a DeepSeek R1 con el mismo prompt. Me respondió los siguientes datos destacables:



Recomendación para Elección de API en tu Laboratorio de Detección de Personas

Basado en tus requisitos y las opciones disponibles, aquí está mi análisis y recomendación:

Análisis Comparativo de APIs

API	Facilidad de Uso	Precisión	Plan Gratuito	Tiempo de Res
Clarifai	Alta	Buena	Generoso	Rápido
Roboflow	Media-Alta	Muy Buena	Limitado	Medio
Google Cloud Vision	Media	Excelente	Créditos limitados	Variable
Azure Computer Vision	Media	Excelente	Créditos limitados	Variable

Reconocimiento de imágenes con Python y Clarifai

Recomendación Principal: Clarifai

Razones para elegir Clarifai para tu laboratorio:

- Facilidad de implementación:**
 - Tiene librerías Python bien documentadas
 - Configuración mínima requerida
 - Ideal para proyectos académicos/laboratorios
- Plan gratuito generoso:**
 - 1,000 operaciones/mes en el plan free
 - Suficiente para desarrollo y pruebas
- Modelos pre-entrenados:**
 - Modelo "general-image-recognition" funciona bien para detección de personas
 - Opción de usar modelos especializados como "person-detection"
- Rápido tiempo de respuesta:**
 - Latencia menor que las soluciones de cloud enterprise
 - Ideal para demostraciones en tiempo real

Finalmente, **elegimos Clarifai**, por su buen equilibrio entre accesibilidad y capacidad de detección. Procedimos a crear la cuenta y mirar la API key que nos dieron.

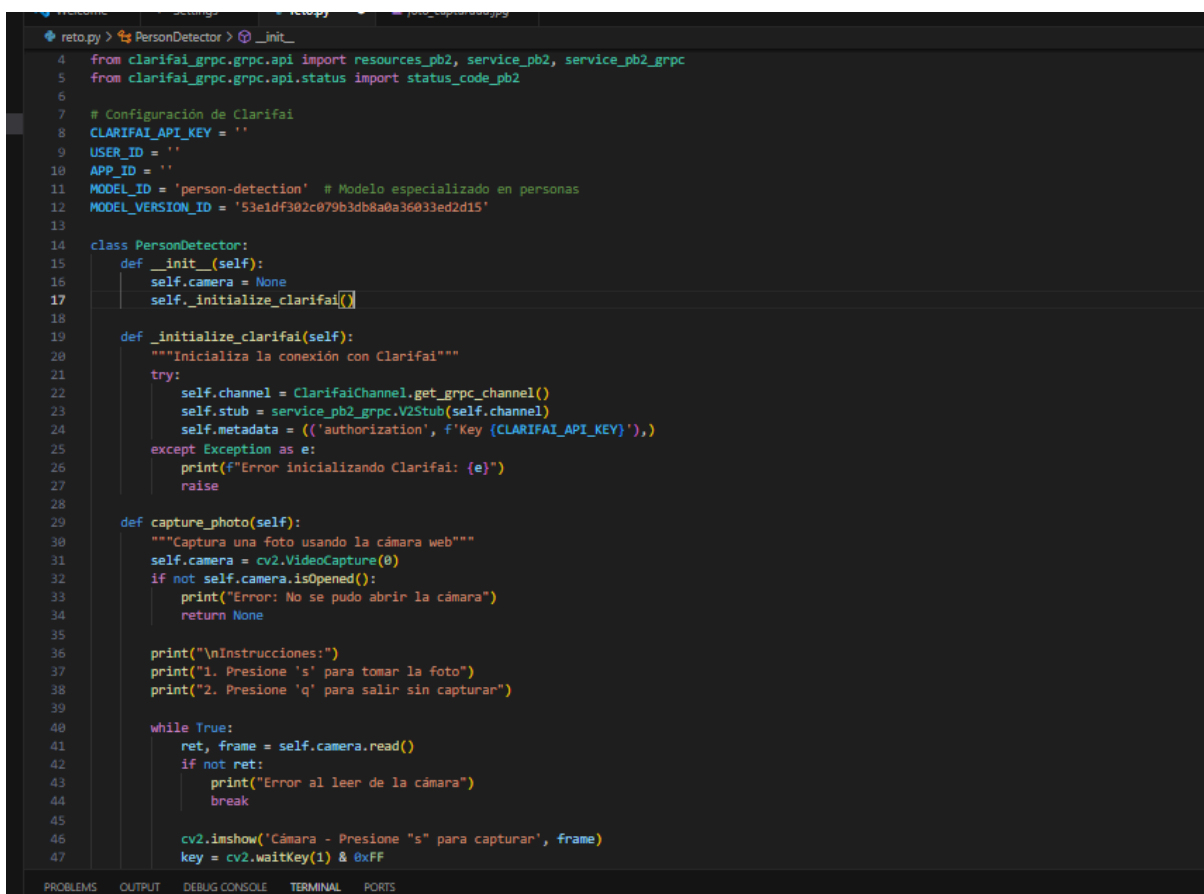
Generando Código del programa

Preguntamos a Deepseek R1 para la realización del código con los requisitos del proyecto. La primera vez que intentamos generar el código usamos el modelo general-image-recognition, este modelo más adelante nos generó complicaciones.

Prompt:

Mira tengo una api key de clarifai con la app y todo quisiera que crearas un código en python donde usé mi cámara para reconocer si hay personas o no hay o tambien otros objetos y que utilices el modelo general-image-recognition

Imagen del código creado por deepseek R1:



```
4 from clarifai_grpc.grpc.api import resources_pb2, service_pb2, service_pb2_grpc
5 from clarifai_grpc.grpc.api.status import status_code_pb2
6
7 # Configuración de Clarifai
8 CLARIFAI_API_KEY = ''
9 USER_ID = ''
10 APP_ID = ''
11 MODEL_ID = 'person-detection' # Modelo especializado en personas
12 MODEL_VERSION_ID = '53e1df382c079b3db8a0a36033ed2d15'
13
14 class PersonDetector:
15     def __init__(self):
16         self.camera = None
17         self.initialize_clarifai()
18
19     def initialize_clarifai(self):
20         """Inicializa la conexión con Clarifai"""
21         try:
22             self.channel = ClarifaiChannel.get_grpc_channel()
23             self.stub = service_pb2_grpc.V2Stub(self.channel)
24             self.metadata = (('authorization', f'Key {CLARIFAI_API_KEY}'),)
25         except Exception as e:
26             print(f"Error inicializando Clarifai: {e}")
27             raise
28
29     def capture_photo(self):
30         """Captura una foto usando la cámara web"""
31         self.camera = cv2.VideoCapture(0)
32         if not self.camera.isOpened():
33             print("Error: No se pudo abrir la cámara")
34             return None
35
36         print("\nInstrucciones:")
37         print("1. Presione 's' para tomar la foto")
38         print("2. Presione 'q' para salir sin capturar")
39
40         while True:
41             ret, frame = self.camera.read()
42             if not ret:
43                 print("Error al leer de la cámara")
44                 break
45
46             cv2.imshow('Cámara - Presione "s" para capturar', frame)
47             key = cv2.waitKey(1) & 0xFF
```

Ahora salió el problema de que no teníamos las librerías necesarias así que las descargamos. Le preguntamos a Chat GPT como instalarlo y nos dijo lo siguiente:

Librerías:

Uso: Captura de imagen desde la cámara web.

- opencv-python

Instalación en cmd: pip install opencv-python

Uso: Envío de solicitudes HTTP a la API REST de Clarifai.

- requests

Instalación en cmd: pip install requests

Uso: Comunicación con la API de Clarifai utilizando gRPC

- clarifai-grpc

Instalación en cmd: pip install clarifai-grpc

Ahora que las librerías ya están descargadas nos salieron algunos inconvenientes con el formato en que se envía los datos a la api con este código. Aunque lo solucionamos por algún motivo nos decía que **no se detectaban personas**.

```
No se detectó una persona en la imagen. Los objetos identificados son:  
- people (confianza: 99.80%)
```

Fue bastante raro porque en el código está diseñado también para mostrar las demás entidades que detectaba, y detectaba people (confianza: 99.80%) y me decía que no detectaba personas

Ya que le pregunte a deepseek r1, ahora le preguntaré a chat GPT para arreglara el código a partir del que deepseek r1 me dio y tambien agregare la información que me dio la terminal

```
Conceptos detectados:  
- portrait (1.00)  
- people (1.00)  
- one (1.00)  
- music (0.99)  
- wear (0.99)  
- adult (0.99)  
- facial expression (0.98)  
- man (0.98)  
- musician (0.97)  
- singer (0.96)  
- retro (0.91)  
- woman (0.89)  
- brunette (0.87)  
- light (0.86)  
- painting (0.86)  
- analogue (0.86)  
- recreation (0.84)  
- art (0.83)  
- vintage (0.82)  
  
No se detectó una persona en la imagen. Los objetos identificados son:  
- portrait (confianza: 99.81%)  
- people (confianza: 99.78%)  
- one (confianza: 99.60%)  
- music (confianza: 99.43%)  
- wear (confianza: 99.13%)  
  
Instrucciones:  
1. Devolver la imagen dentro de <img>
```

y tambien le di este prompt extra:

mira estoy en la cámara y no me detecta {La imagen anterior}

Gracias por mostrar el resultado. En efecto, el modelo está reconociendo claramente que hay personas en la imagen (conceptos como *people*, *man*, *adult*, *boy* todos con >95%), pero no está retornando el concepto exacto "person" que tu código busca para marcarlo como detección humana.

¿Qué pasa?

En tu código actual, estás buscando exactamente esto:

```
python
if concept.name == 'person' and concept.value > 0.95:
    es_persona = True
```

Pero como viste, "person" no aparece. En cambio, aparecen sinónimos o categorías similares como:

- `people`
- `man`
- `adult`
- `boy`

✓ **Solución:**

Vamos a ampliar la lógica para que detecte personas si encuentra cualquiera de esos términos relacionados.

Reemplaza esto:

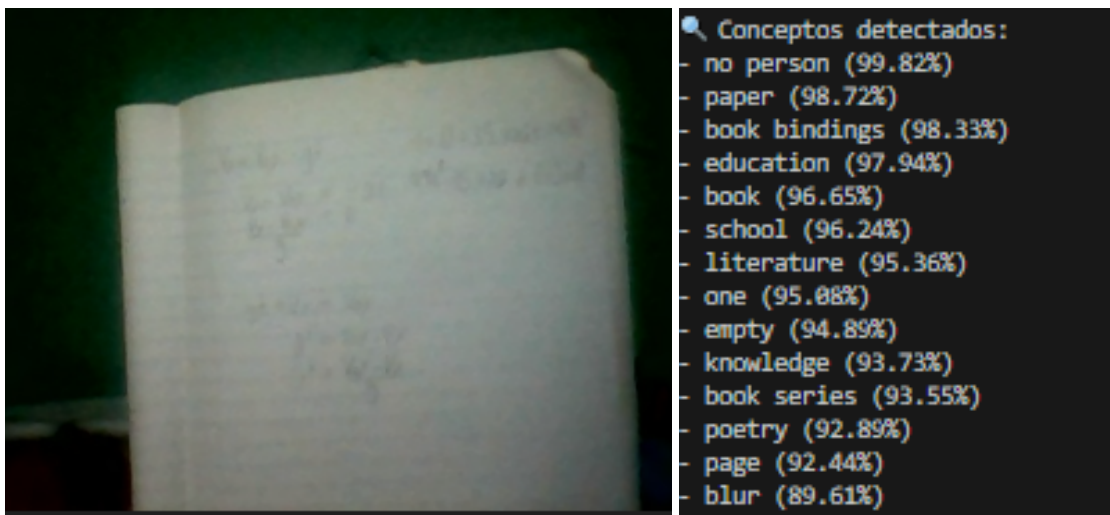
```
python
if concept.name == 'person' and concept.value > 0.95:
    es_persona = True
```

Por esto:

```
python
if concept.name in ['person', 'people', 'man', 'woman', 'adult', 'boy', 'girl'] and c
    es_persona = True
```

Corregí el código con la solución que me dio Chat GPT.

Ahora como experimento mostre solo un cuaderno a la cámara par ver si detecta el objeto o si detecta personas y me dio lo siguiente:



```
✗ No se detectó una persona en la imagen.

▣ Lista de objetos detectados (mayor a 70% de confianza):
◆ no person: 99.82%
◆ paper: 98.72%
◆ book bindings: 98.33%
◆ education: 97.94%
◆ book: 96.65%
◆ school: 96.24%
◆ literature: 95.36%
◆ one: 95.08%
◆ empty: 94.89%
◆ knowledge: 93.73%
```

Luego me puse un compañero en la cámara y nos dio lo siguiente:

```
✔ ¡Se ha detectado una persona en la imagen!

▣ Lista de objetos detectados (mayor a 70% de confianza):
▶ portrait: 99.67%
▶ people: 99.31%
▶ one: 98.72%
▶ adult: 98.38%
▶ wear: 96.91%
▶ man: 96.59%
▶ street: 95.24%
▶ light: 94.68%
▶ outdoors: 92.31%
```

En conclusión encontramos un código que funcionó correctamente, y cumplía con todos los requisitos y también le pedimos a Chat GPT que nos proporcionará un informe del código para poder tener una idea de lo que hace en el siguiente link:

https://drive.google.com/file/d/1663j0JFizCsZNdotvIk7n83A3P1yD_hT/view?usp=drive_link

SEGUNDO CÓDIGO:

la elaboración del código no lo estábamos haciendo varios para resolver los problemas sobre la conectividad de la API key y otras cosas. Como lo había mencionado antes tuvimos problemas al reconocer las personas. Mi compañero Lester tuvo el siguiente problema: El código le funcionaba de manera correcta cuando él estaba frente de la cámara, le decía la terminal que se detectó una persona, pero cuando él se quitó de la cámara todavía le detectaba.

Estuvimos investigando y pensábamos que la raíz del problema sería el modelo `general-image-recognition` pensábamos que este modelo no detectaba bien las personas de la manera mas eficiente, así que mi compañero lester uso el modelo **face-detection**.

Consiguió que le funcionara el código, además ahora puede detectar si hay varias personas en la webcam.

```
14 def capturar_imagen():
15     """Captura una imagen utilizando la cámara y la guarda como captura.jpg."""
16     camara = cv2.VideoCapture(0)
17     if not camara.isOpened():
18         print("X No se pudo acceder a la cámara.")
19         return None
20
21     print("⏏ Presiona 's' para tomar la foto o 'q' para salir")
22
23     while True:
24         ret, frame = camara.read()
25         if not ret:
26             print("X Error al capturar el frame.")
27             break
28
29         cv2.imshow('Cámara - Presiona "s" para tomar foto', frame)
30         key = cv2.waitKey(1)
31         if key == ord('s'):
32             filename = 'captura.jpg'
33             cv2.imwrite(filename, frame)
34             print(f"✅ Imagen guardada como {filename}")
35             break
36         elif key == ord('q'):
37             print("X Saliendo sin tomar foto.")
38             filename = None
39             break
40
41     camara.release()
42     cv2.destroyAllWindows()
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
✅ ¡Se han detectado 4 persona(s) en la imagen!
PS C:\Users\HP\Desktop\reto> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/HP/Desktop/reto/reto.py
⏏ Presiona 's' para tomar la foto o 'q' para salir
✅ Imagen guardada como captura.jpg

⚡ Analizando imagen...
✅ Predicción realizada con éxito!

✅ ¡Se han detectado 3 persona(s) en la imagen!
PS C:\Users\HP\Desktop\reto> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/HP/Desktop/reto/reto.py
⏏ Presiona 's' para tomar la foto o 'q' para salir
✅ Imagen guardada como captura.jpg

⚡ Analizando imagen...
✅ Predicción realizada con éxito!

✅ ¡Se han detectado 5 persona(s) en la imagen!
PS C:\Users\HP\Desktop\reto> |
```

El código de Lester funcionaba así que ya tenemos otro potencial código del proyecto que funciona correctamente.

Informe creado por Chat GPT para analizar el segundo código para que nuestro grupo lo entienda.

Link: https://drive.google.com/file/d/1njeNzINkKaVeEbZkLXJD7BgIRC-fEs2D/view?usp=drive_link

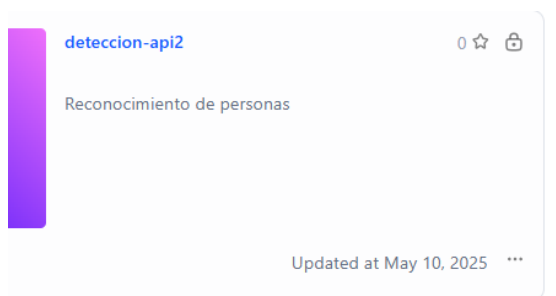
Ahora con los dos códigos hechos correctos en conclusión aprendimos que en la elaboración del código del proyecto, los ILM que preguntamos hacían códigos de manera distinta y generaban códigos que tenían error o resultados insatisfactorios. Al final logramos generar los códigos correctos complementados con las dos ILM principales que fueron Chat GPT y DeepSeek r1.

Cronología de Problemas encontrados al realizar el proyecto

El primer problema fue que en clase como todos teníamos que descargar Python en VS code para hacer el proyecto, esto generaría problemas en la velocidad de la red, lo cual nos atrasa considerablemente.

Otro problema añadido del anterior fue la descarga de librerías especializadas, al principio micompañero encontró problemas instalando las librerías y esto nos consumió algo de tiempo.

el tercer problema surge cuando probamos el código y nos daba inconsistencias con la API key y el formato con el que se enviaba la imagen al servidor, Lester arreglo eso al sugerir a los demás integrantes que nos faltaba crear la app en Clarifai.



también más tarde se averiguó que no hacía falta crear la app sino que se podría usar lo siguiente:

User ID: clarifai

App ID: main

Model ID: (por ejemplo) face-detection o general-image-recognition

Model Version ID: (opcional, pero recomendable para mayor precisión)

API Key: La tuya personal, generada desde tu cuenta.

Luego surgió el problema que más nos complicó y es el error de que mi compañero Lester no le detectaba bien si había personas o no, usando el modelo general ("general-image-recognition"), la etiqueta "person" aparecía con una confianza alta (0.96) incluso cuando no había una persona real en la imagen.

Esto nos llevó a considerar cambiar de modelo. Ya concluida el trabajo del proyecto averiguamos que el problema surgió de los falsos positivos causados por la iluminación, la calidad de la cámara y por último como lo solucionamos es generando otro código por si había un error de semántica donde hacía que el código generaba resultados incorrectos.

A continuación mostraré los modelos que usábamos y lo que pensábamos antes de finalizar el proyecto

Modelos usados

- **Modelo general de Clarifai:** `general-image-recognition`
 - Problema: Detección poco precisa, tendía a detectar "person" en todo.
- **Modelo de detección de rostro:** `face-detection`
 - Ventaja: Más específico, nos permitió contar mejor cuántas personas aparecían en la imagen.

Para finalizar la cronología de los problemas nuestro grupo al hacer los dos códigos nos dimos cuenta que tuvo mucho que ver la semántica de los códigos sobretodo el de `general-image-recognition`, que pensábamos que el model no funciona muy bien y el principal problema era cómo estaba hecho el código.

Conclusión

Lo que notamos en nuestro grupo es que aunque sabíamos que este laboratorio su objetivo era la de usar inteligencia artificial para elaborar un proyecto sin tener el conocimiento de el lenguaje Python o usar API gratuitas, y como gracias a las YA podemos crear proyectos con cierta complejidad, con el tiempo invertido adecuado y la generación de prompts eficientes. Notamos que si tuviéramos el conocimiento adecuado esta tarea seria aun mas eficiente y productiva, porque tuvimos problemas de semántica entre otros.