

1. BREVE DESCRIPCION DEL PROYECTO

El proyecto se enfocó en el desarrollo de una aplicación de computadora con capacidades avanzadas de visión artificial para reconocer y clasificar frutas, evaluando su estado de madurez, y determinando si están en condiciones saludables o en estado de descomposición. La aplicación también incorpora la capacidad de identificar y extraer información crucial sobre la fecha de vencimiento de productos envasados a través del escaneo de códigos de barras o el reconocimiento de fechas impresas en los envases.

2. INTRODUCCION

En nuestro mundo cada vez más dependiente de la tecnología, la innovación en el ámbito de la alimentación y la seguridad del consumidor se vuelve de vital importancia. El proyecto que se presenta a continuación aborda directamente estos desafíos mediante el desarrollo de un software de computadora con capacidades de visión artificial. El enfoque principal de esta aplicación es el reconocimiento preciso de frutas y la evaluación de su estado, diferenciando entre productos en condiciones óptimas y aquellos que han alcanzado un estado de descomposición.

La aplicación no solo se limita a la clasificación visual de las frutas, sino que también incorpora una funcionalidad esencial para la salud y seguridad del consumidor: la identificación automática de fechas de vencimiento. Esta capacidad se logra mediante la lectura de códigos de barras y el reconocimiento de fechas impresas en los envases, proporcionando a los usuarios información crítica sobre la frescura y la vigencia de los productos.

El proyecto responde a la creciente demanda de soluciones tecnológicas en el sector alimentario, desarrollando un software que tiene el potencial de transformar significativamente la manera en que evaluamos y gestionamos la frescura de los alimentos, promoviendo así una experiencia más informada y segura para los consumidores

3. OBJETIVO

Como objetivo principal nos dimos la tarea de crear un software de computadora que sea capaz de identificar que alimentos son aptos para el consumo humano sin llevar ningún riesgo al consumidor. Aplicando como herramienta principal las técnicas de visión artificial aprendidas a lo largo del semestre

4. MARCO TEORICO

Los conceptos utilizados para realizar este proyecto que fueron aquellos aprendidos y estudiados a lo largo del semestre son:

- **Visión Artificial:**

La visión artificial es un campo de la inteligencia artificial que se centra en la capacitación de sistemas informáticos para interpretar y comprender el mundo visual de la misma manera que lo hacen los seres humanos. Este enfoque implica el uso de algoritmos y modelos de aprendizaje profundo para analizar y extraer información útil de imágenes y videos.

- **Redes Neuronales Convolucionales (CNN):**

Convolutional Neural Networks (CNN), o Redes Neuronales Convolucionales en español, son un tipo de red neuronal artificial especializada en el reconocimiento de patrones en datos de tipo visual, como imágenes y videos. Estas redes han revolucionado la visión por computadora y la percepción de máquinas al tratar de emular el comportamiento de la corteza visual del cerebro humano.

- **Detección de Objetos:**

La detección de objetos se refiere a la capacidad de identificar y ubicar objetos específicos en una imagen. En nuestro caso, implica la identificación precisa de frutas dentro de una escena, facilitada por algoritmos especializados como los utilizados en Faster R-CNN (Region-based Convolutional Neural Network).

- **Lectura de Códigos de Barras:**

La lectura de códigos de barras es una tecnología establecida para la identificación única de productos. Utiliza técnicas de procesamiento de imágenes y reconocimiento de patrones para interpretar la información codificada en el código de barras, incluida la fecha de vencimiento.

- **Procesamiento de Lenguaje Natural (PLN):**

El procesamiento del lenguaje natural (PLN), es una rama de la inteligencia artificial que permite a las computadoras comprender, generar y manipular el lenguaje humano. El procesamiento del lenguaje natural tiene la capacidad de interrogar los datos con el texto o la voz del lenguaje natural.

- **Aprendizaje Supervisado:**

En este enfoque, los algoritmos se entrenan en un conjunto de datos etiquetado que contiene ejemplos de entrada y la salida deseada. El algoritmo aprende a mapear las entradas a las salidas correspondientes, lo que permite hacer predicciones en nuevos datos no etiquetados.

5. METODOLOGIA

En la metodología del desarrollo del software utilizando como herramienta principal la visión artificial seguimos una metodología convencional que consistió en:

1. Adquisición y Preparación de Datos:

- **Obtención de Conjunto de Datos:** Se recolectaron imágenes de frutas en diversas condiciones y productos enlatados con códigos de barras para crear un conjunto de datos representativo haciendo uso de la herramienta **OPEN IMAGES DATASET V7**, que nos proporcionó los conjuntos de datos adecuados y ya etiquetados

para el entrenamiento de nuestros modelos.



- **División del Conjunto de Datos:** Se dividió el conjunto de datos en conjuntos de entrenamiento y prueba para evaluar la capacidad del modelo para generalizar.

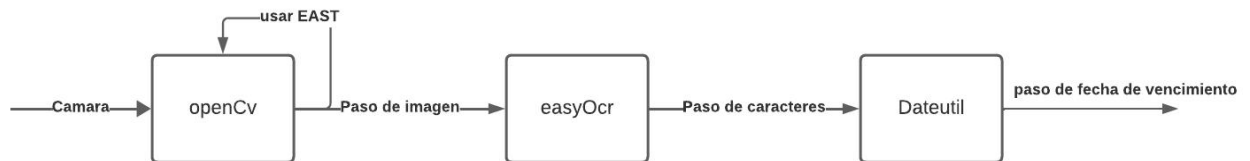
2. Selección del Modelo Preentrenado:

Por falta de recursos para lograr entrenar un modelo desde 0 optamos por la opción de usar un modelo preentrenado con capacidades de

detección de objetos y reconocimiento de texto, como un modelo **YOLO** preentrenado para la detección de frutas y un modelo de reconocimiento de texto preentrenado compatible **EasyOCR**.

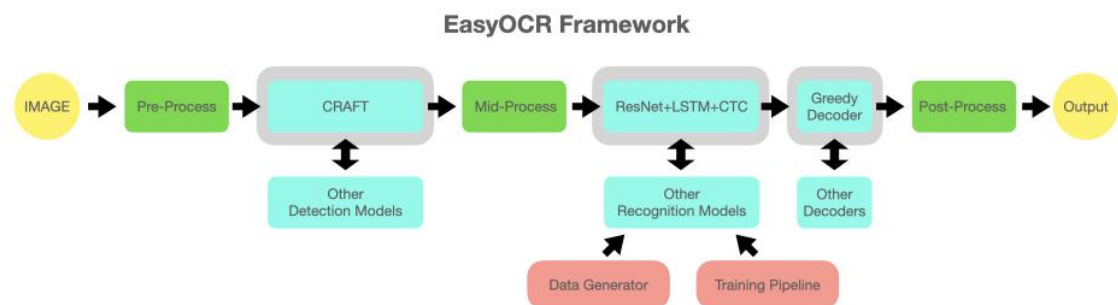
3. Procesamiento de Imágenes y reconocimiento de Texto y Fechas:

Modelo:

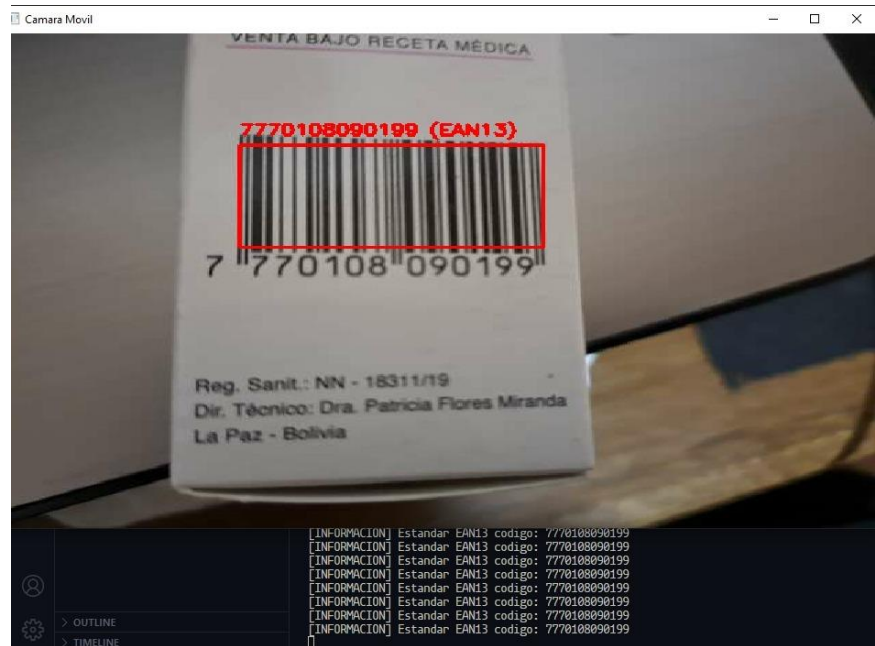


- **Preprocesamiento de Imágenes:** OpenCV se aplicó para realizar operaciones de preprocesamiento, como ajustes de tamaño, normalización y mejora del contraste.
- **EasyOCR para Extracción de Texto:** Se empleó EasyOCR para extraer texto de las imágenes, focalizándose en las fechas impresas en los envases.

Siguiendo este modelo se observa que easyocr nos procesa varias veces la imagen para lograr la mayor fidelidad posible del modelo



- **ZBar para Lectura de Códigos de Barras:** ZBar Barcode Reader se utilizó para la lectura de códigos de barras en productos, permitiendo la obtención de información, incluidas las fechas de vencimiento.



4. Procesamiento de Fechas y Evaluación del Estado de Frutas:

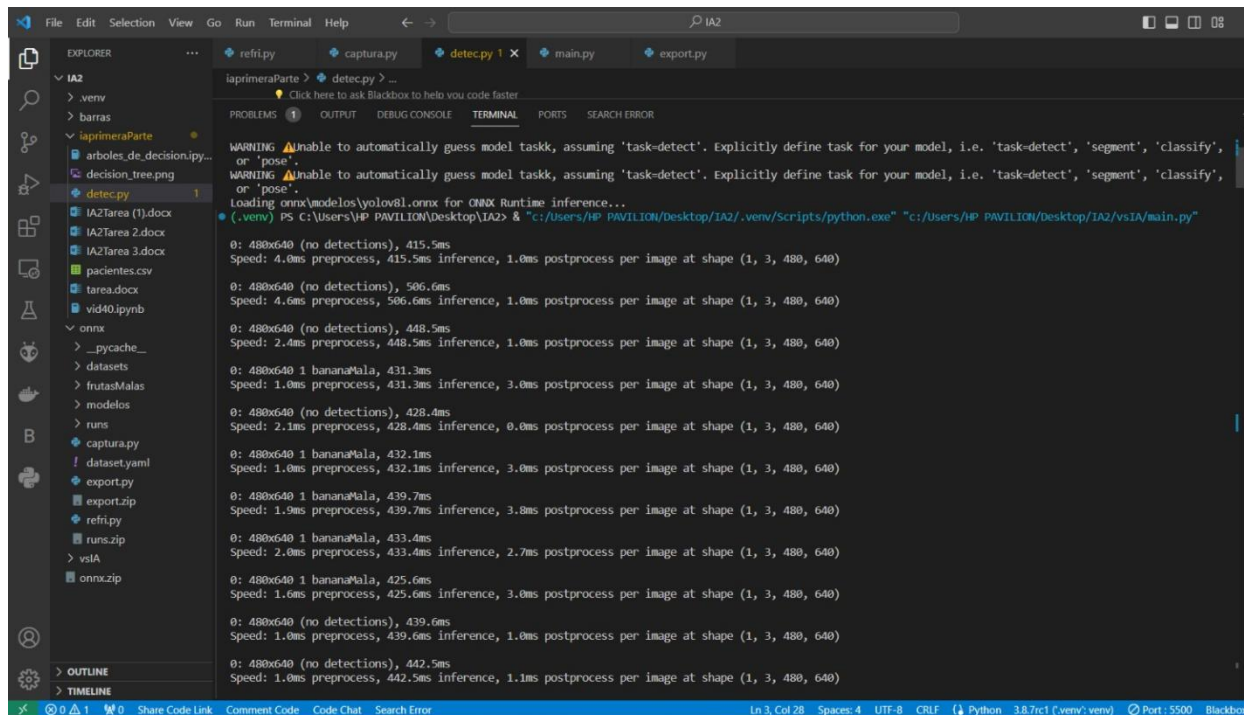
- **Manipulación de Fechas:** Se utilizó la librería de Python **DATEUTIL**, se utilizó para analizar y manipular las fechas extraídas, facilitando la comparación con la fecha actual y la evaluación del estado de frescura de los productos.



- **Operaciones Numéricas:** **NUMPY** se integró para realizar cálculos numéricos relacionados con la evaluación del porcentaje de descomposición de las frutas.

5. Integración de Modelos ONNX:

- **Compatibilidad de Modelos con ONNX:** Se garantizó que los modelos utilizados fueran compatibles con el formato ONNX para facilitar la interoperabilidad y la implementación en diferentes frameworks de aprendizaje profundo.



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a terminal on the right. The file explorer shows a project structure with folders like 'IA2', 'venv', and 'barra'. The terminal displays the output of a Python script named 'detec.py'. The script is running on a Windows machine (C:\Users\HP PAVILION\Desktop\IA2) and is using the ONNX runtime for inference. The output shows several warnings about the model task and the inference speed for different images.

```
WARNING: Unable to automatically guess model task, assuming 'task-detect'. Explicitly define task for your model, i.e. 'task-detect', 'segment', 'classify', or 'pose'.
WARNING: Unable to automatically guess model task, assuming 'task-detect'. Explicitly define task for your model, i.e. 'task-detect', 'segment', 'classify', or 'pose'.
Loading onnx\models\yolov8l.onnx for ONNX Runtime inference...
(.venv) PS C:\Users\HP PAVILION\Desktop\IA2> & "c:/Users/HP PAVILION/Desktop/IA2/.venv/Scripts/python.exe" "c:/Users/HP PAVILION/Desktop/IA2/vsIA/main.py"

0: 480x640 (no detections), 415.5ms
Speed: 4.0ms preprocess, 415.5ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 506.6ms
Speed: 4.6ms preprocess, 506.6ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 448.5ms
Speed: 2.4ms preprocess, 448.5ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 bananaMala, 431.3ms
Speed: 1.0ms preprocess, 431.3ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 428.4ms
Speed: 2.1ms preprocess, 428.4ms inference, 0.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 bananaMala, 432.1ms
Speed: 1.0ms preprocess, 432.1ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 bananaMala, 439.7ms
Speed: 1.9ms preprocess, 439.7ms inference, 3.8ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 bananaMala, 433.4ms
Speed: 2.0ms preprocess, 433.4ms inference, 2.7ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 bananaMala, 425.6ms
Speed: 1.6ms preprocess, 425.6ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 439.6ms
Speed: 1.0ms preprocess, 439.6ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 (no detections), 442.5ms
Speed: 1.0ms preprocess, 442.5ms inference, 1.1ms postprocess per image at shape (1, 3, 480, 640)
```

6. Entrenamiento del Modelo

- Es sabido que utilizamos un modelo preentrenado para poder ahorrar en recursos a la hora de crear el software debido a que no contamos con los recursos deseados para poder entrenarlos nosotros mismos, sin embargo eso no quiere decirle que no lo entrenemos, el entrenamiento del modelo es fundamental para que el software funcione correctamente:

- Entrenamiento solo con imágenes:

```

vsIA > / dataset.yaml
1 Click here to ask Blackbox to help you code faster
2 train: C:\Users\VP\ PAVILION\Desktop\IA2\vsIA\datasets\data\train
3 val: C:\Users\VP\ PAVILION\Desktop\IA2\vsIA\datasets\data\val
4 test:
5 names: ["manzana", "banana"]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERRORS
optimizer: 'optimizer-auto' found, ignoring 'lr=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr' and 'momentum' automatically...
optimizer: AdamW(lr=0.001667, momentum=0.9) with parameter groups 86 weight(decay=0.0), 97 weight(decay=0.0005), 96 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\segment\train4
Starting training for 60 epochs...

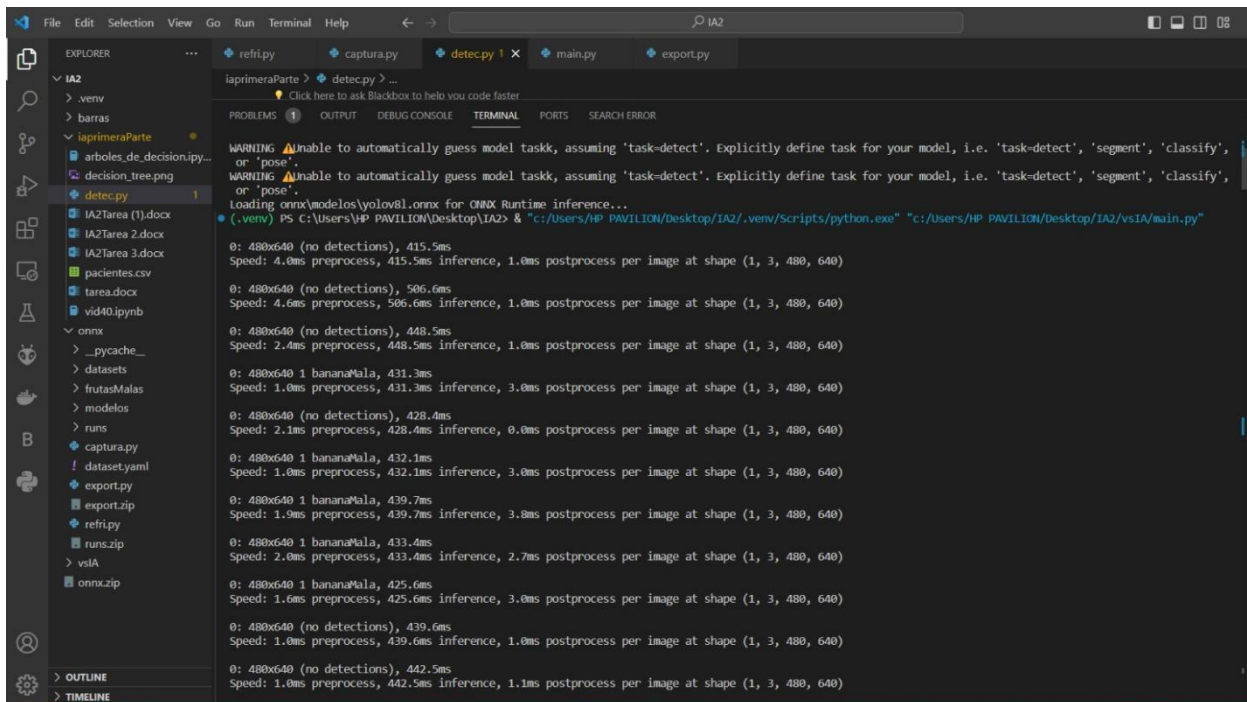
Epoch GPU_mem box_loss seg_loss cls_loss dfl_loss Instances Size
1/11 [00:03:00] 1/60 640: 0% | 0/11 [00:03:00] 1/60 0% 0.6151 3.66 2.541 1.195 6 640: 95% | 1/11 [00:07:00] 1/60 0% 0.5
42 3.409 2.743 1.179 4 640: 18% | 2/11 [00:07:00] 1/60 0% 0.6228 3.664 2.81 1 1/60 0% 0.6372 3.228 2.691 4
1.243 640: 18% | 2/11 [00:09:00] 1/60 0% 0.6228 3.664 2.81 1 1/60 0% 0.6372 3.228 2.691 4
7/11 [00:16:00] 1/60 0% 0.6308 2.879 2.473 1.244 7 640: 45% | 5/11 [00:16:00] 1/60 0% 0.7219 2.76 2.68 1.268 4
9 2.76 2.68 1.268 4 640: 45% | 5/11 [00:19:00] 1/60 0% 0.7874 3.552 2.637 1.387 4 640: 55% | 6/11 [00:22:00] 1/60 0% 0.7497 3.293
1/60 0% 0.7874 3.552 2.637 1.387 4 640: 64% | 7/11 [00:25:00] 1/60 0% 0.7497 3.293
2.522 1.293 0 1/60 0% 0.7217 3.069 2.535 1.258 5 640: 82% | 9/11 [00:28:00] 1/60 0% 0.711 2.905 2.458 1.2
0 0.7217 3.069 2.535 1.258 5 640: 82% | 9/11 [00:31:00] 1/60 0% 0.711 2.905 2.458 1.2
51 6 640: 82% | 9/11 [00:31:00] 1/60 0% 0.711 2.905 2.458 1.2
11 [00:31:00] 1/60 0% 0.7444 2.932 2.414 1.259 4 640: 91% | 10/11 [00:33:00] 1/60 0% 0.7444
2.932 2.414 1.259 4 640: 100% | 11/11 [00:33:00:00] 3.055/it | 11/11 [00:33:00] 1/60 0% 0.7444 2.932 2.414 1.259 4
Class Images Instances box(p
all 21 21 0.6 0.866 0.818 0.729 0.598 0.869 0.786 0.698

Epoch GPU_mem box_loss seg_loss cls_loss dfl_loss Instances Size
2/60 0% 0.6282 1.349 1.647
Class Images Instances box(p
all 21 21 0.68 0.935 0.82 0.699 0.68 0.935 0.818 0.723

Epoch GPU_mem box_loss seg_loss cls_loss dfl_loss Instances Size
3/60 0% 0.8756 1.384 2.292

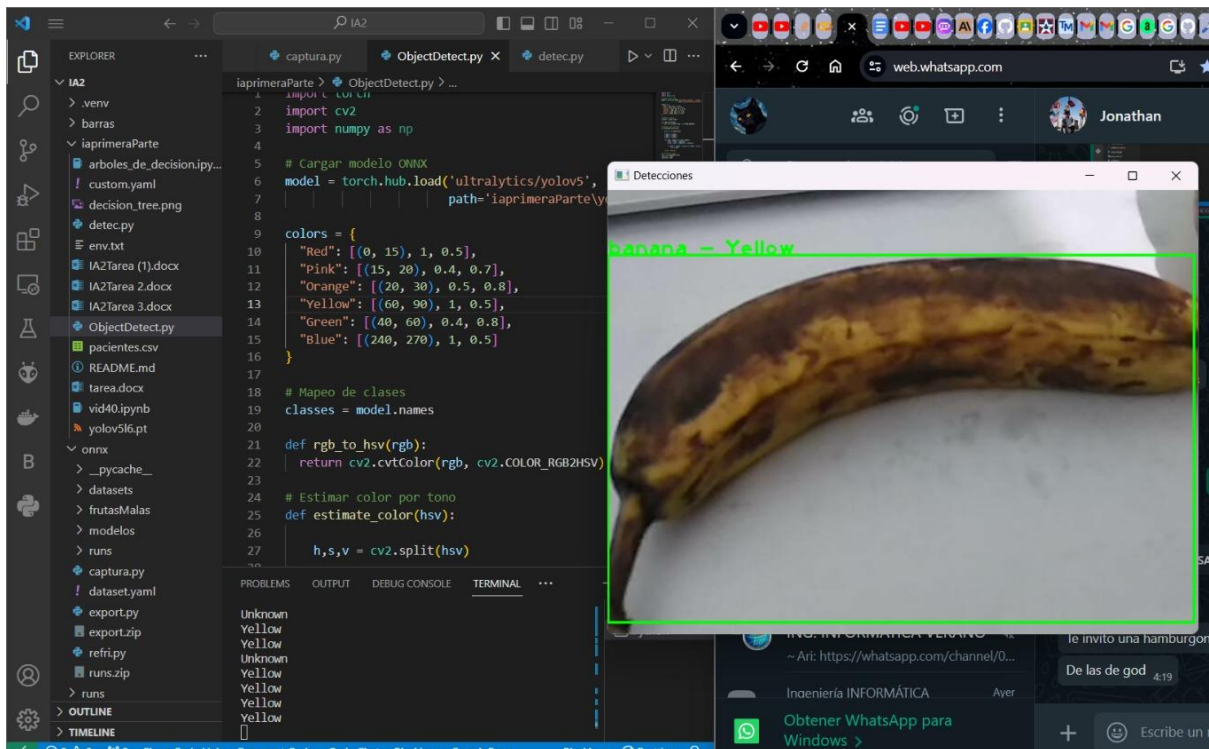
```

- Entrenamiento con imágenes y labels.



7. Pruebas y Evaluación:

- Realizar pruebas exhaustivas utilizando el nuevo conjunto de datos.



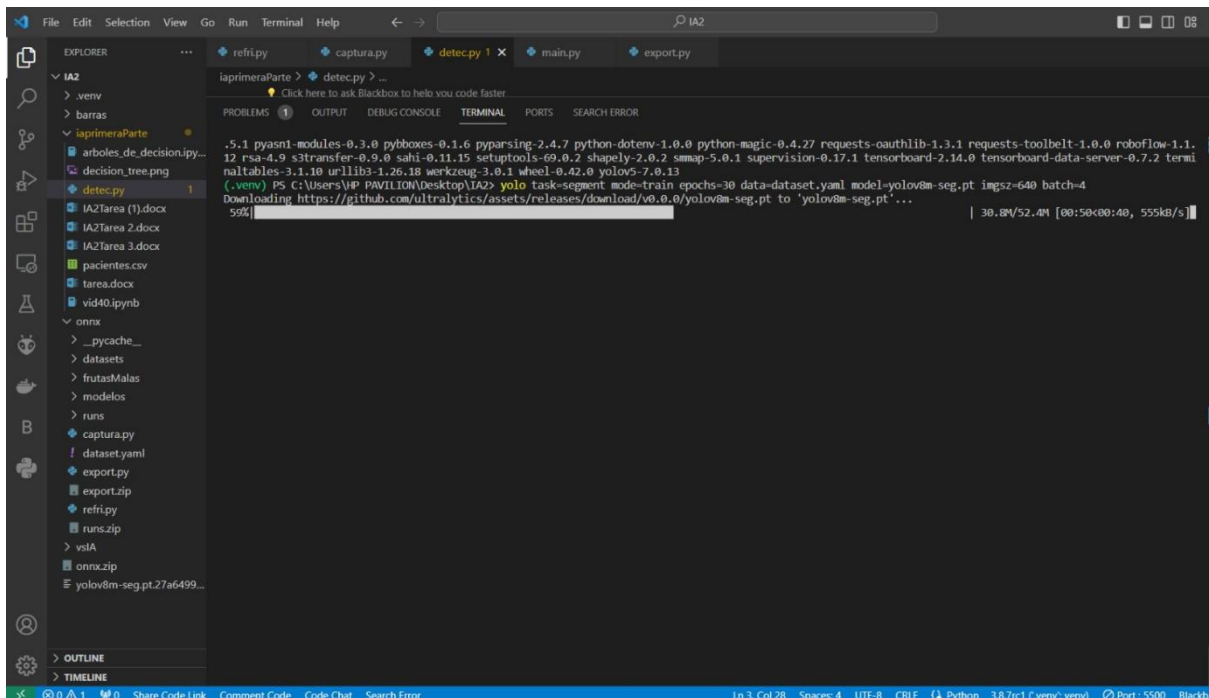
8. Integración de la Aplicación:

- Desarrollar la interfaz de usuario teniendo en cuenta los resultados del modelo adaptado.
- Integrar todos los componentes para lograr una aplicación cohesiva y funcional.

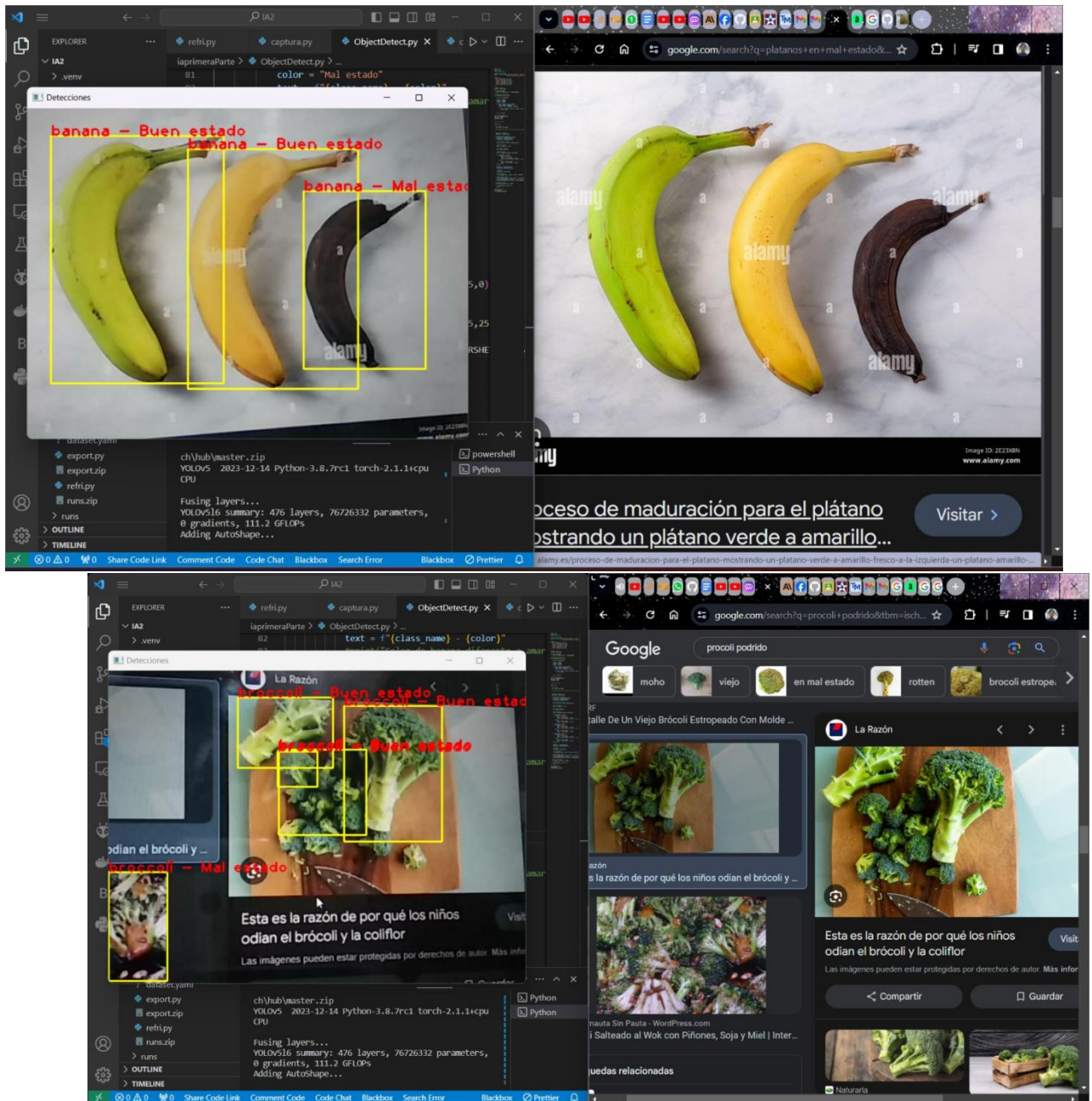


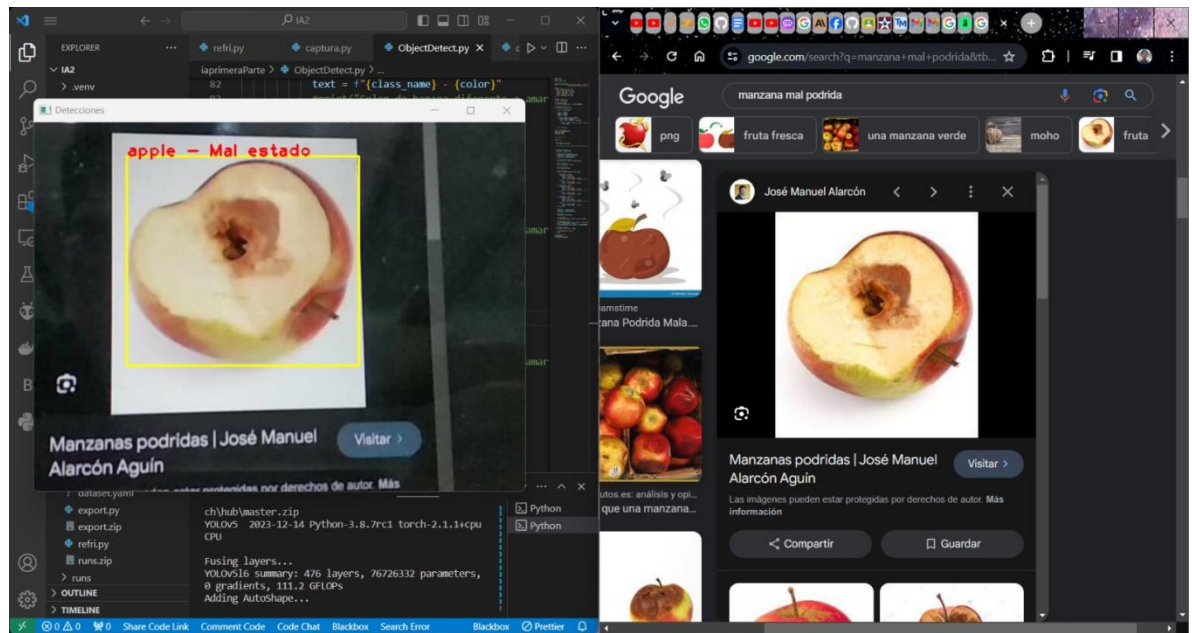
9. Ajuste Iterativo:

- Realizar ajustes iterativos en la adaptación del modelo en función de los resultados obtenidos durante las pruebas.



- Refinar el modelo y la aplicación para mejorar la precisión y la eficiencia.





6. ALGORITMO DE VISION ARTIFICIAL UTILIZADO (YOLO)

Algunos de los elementos clave del algoritmo YOLOv4:

1. Backbone Network:

- YOLOv4 utiliza una red neuronal profunda como espinazo (backbone). En particular, se basa en CSPDarknet53, que es una variante del modelo Darknet, conocido por su eficiencia en términos de cómputo y capacidad para capturar características relevantes.

2. Feature Pyramid Network (FPN):

- Se implementa FPN para capturar características a diferentes escalas espaciales, lo que mejora la capacidad del modelo para detectar objetos de diferentes tamaños.

3. Neck Architecture:

- YOLOv4 utiliza un diseño de "neck" (cuello) que incluye bloques PANet (Path Aggregation Network) y SAM (Spatial Attention Module). Estos módulos ayudan a mejorar la información espacial y la agregación de características.

4. Detección en Varios Niveles (YOLO Head):

- La cabeza de detección YOLO (YOLO head) se encarga de predecir las cajas delimitadoras y las clases de objetos en varios niveles de la pirámide de características generada por la red.

5. Anchor Boxes y Predicciones:

- YOLOv4 utiliza "anchor boxes" para mejorar la precisión de las predicciones. Las predicciones incluyen información sobre la posición (coordenadas de la caja delimitadora), clase y confianza de cada objeto detectado.

6. Activación Leaky ReLU y Batch Normalization:

- Se utilizan funciones de activación Leaky ReLU para introducir no linealidades en la red, y la normalización por lotes (Batch Normalization) para estabilizar y acelerar el entrenamiento.

7. Optimización y Entrenamiento:

- El algoritmo YOLOv4 utiliza técnicas de optimización, como el optimizador Adam, y se entrena con grandes conjuntos de datos anotados para aprender a detectar una amplia variedad de objetos.

8. Hardware Acceleration:

- YOLOv4 se ha diseñado para aprovechar la aceleración de hardware, como GPU (Unidades de Procesamiento Gráfico), para realizar inferencias más rápidas y eficientes.

7. REFERENCIAS:

- "Machine Learning - Redes Convolucionales" por: Paco Llalli Judith Margarita, Perez Valenzuela Josue Jonathan, Valdivia Viscarra Nicolás Mateo

- "Visión Artificial" por: Calli Rodriguez Boris Victor, Fernandez Vallejos Jose Franklim, Navia Peña Samuel Douglas, Peralta Flores Jonathan
- "TEORÍA: PROCESAMIENTO DE LENGUAJE NATURAL" por: Alvarez Rojas Alexander James, Calle Pinto William, Flores Burgoa Dorian Joaquin, Jimenez Terceros Elmer
- "YOLOv4" por: [FastSAM \(Fast Segment Anything Model\) - Ultralytics YOLOv8 Docs](#)
- "Open Images v7" por: [Open Images V7 \(storage.googleapis.com\)](#)