UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Mecatrónica

SISTEMA DE CONTEO DE CAJAS DE CARTÓN Y BOTELLAS BASADO EN VISIÓN ARTIFICIAL PARA TIENDAS DE VÍVERES

Trabajo de grado previo a la obtención del título de Ingeniero en Mecatrónica

Autor:

Cristian David Calle Cañar

Director:

Ing. Carlos Xavier Rosero Chandi, PhD.

Ibarra – Ecuador

2024

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	172531227-4		
APELLIDOS Y NOMBRES:	Calle Cañar Cristian David		
DIRECCIÓN:	Vicente Estrella y Entrada a la UCCOPEM, Tabacundo		
EMAIL:	cdcallec@utn.edu.ec		
TELÉFONO FIJO:	-	**TELÉFONO MÓVIL:**	0962545452

DATOS DE LA OBRA	
TÍTULO:	Sistema de conteo de cajas de cartón y botellas basado en visión artificial para tiendas de víveres.
AUTOR (ES):	Calle Cañar Cristian David
FECHA: DD/MM/AAAA	05/08/2024
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **GRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
ASESOR /DIRECTOR:	Ing. Xavier Rosero, PhD, - Ing. Cosme Mejía, MSc.

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 5 de agosto de 2024

EL AUTOR:

(Firma).....

Nombre: Cristian David Calle Cañar

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 5 de agosto de 2024

Ing. Carlos Xavier Rosero Chandi, PhD.

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de titulación, el mismo que se ajusta a las normas vigentes de la Unidad Académica de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

(f) .....

Ing. Carlos Xavier Rosero Chandi, PhD.

C.C.:1002515821

APROBACIÓN DEL COMITÉ CALIFICADOR

El Tribunal Examinador del trabajo de titulación "SISTEMA DE CONTEO DE CAJAS DE CARTÓN Y BOTELLAS BASADO EN VISIÓN ARTIFICIAL PARA TIENDAS DE VÍVERES" elaborado por Cristian David Calle Cañar, previo a la obtención del título del Ingeniero en Mecatrónica, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

(f) .....
Ing. Carlos Xavier Rosero Chandi, PhD.
C.C.:1002515821

(f) .....
Ing. Cosme Damián Mejía Echeverria, MSc.
C.C.:1002641288

Dedicatoria

Dedico este trabajo de titulación a mi madre y abuela, por haberme guiado y educado con amor, permitiéndome alcanzar estos logros académicos, a mi familia, en especial a mis tíos y primo, cuyo apoyo y motivación fueron clave en este proceso académico.

Cristian

Agradecimiento

Agradezco profundamente a Dios por las bendiciones recibidas durante mi camino académico. También agradezco a mi madre y familia, quienes me incentivaron a superarme personalmente.

A mis amigos por amenizar el transcurso de esta etapa universitaria, a través de la motivación y las lecciones aprendidas en los proyectos realizados.

Al Dr. Xavier Rosero y al MSc. Cosme Mejía por su orientación en la realización de este trabajo, así como también por demostrar calidez humana como docentes en la carrera.

Cristian

Índice general

Índice de figuras

Índice de tablas

Resumen

Hoy en día existen múltiples aplicaciones en las cuales se implementan sistemas de visión artificial que se enfocan en automatizar procesos y facilitar el trabajo a las personas en ciertas actividades, como es el caso del conteo de objetos. Esta actividad en particular se realiza regularmente en las tiendas de víveres, aunque de manera manual trayendo como consecuencia el empleo adicional de tiempo.

En respuesta a esta necesidad, se plantea un sistema basado en visión artificial que permite el conteo y registro de botellas y cajas de cartón, que ingresan diariamente al establecimiento. Consta de una interfaz optimizada para computadora creada en Python, en donde el usuario puede contar, registrar y visualizar los datos almacenados de botellas de plástico y cajas de cartón al presionar un botón. Para la adquisición de video se emplea una cámara que está orientada hacia un espacio destinado a la recepción de productos.

El reconocimiento de los objetos se realiza a través de la implementación de la red neuronal convolucional SSD MobileNet V2 entrenada mediante el framework de TensorFlow. El conjunto de datos empleado contine imágenes de botellas de agua y gaseosas, y cajas de cartón. La estrategia utilizada para el conteo de las botellas se basa en la identificación de las tapas, debido a que estas son más notorias, optimizando así el proceso. Los resultados de este trabajo muestran que el sistema diseñado facilita el ingreso de productos a través de las distintas funcionalidades implementadas. El entrenamiento de la red de detección de objetos condujo a una precisión general de 90.8%, evaluada mediante una matriz de confusión.

Palabras clave: Visión artificial, Detección de objetos, Interfaz en Python, TensorFlow, Tiendas de víveres.

Abstract

Today there are multiple applications in which machine vision systems are implemented that focus on automating processes and facilitating the work of people in certain activities, such as counting objects. This activity in particular is regularly performed in grocery stores, although manually bringing as a consequence the additional use of time.

In response to this need, we propose a system based on artificial vision that allows the counting and registration of bottles and cartons, which enter the store daily. It consists of a computer-optimized interface created in Python, where the user can count, record and visualize the stored data of plastic bottles and cardboard boxes at the push of a button. For video acquisition, a camera is used that is oriented towards a space intended for receiving products.

Object recognition is performed through the implementation of the SSD MobileNet V2 convolutional neural network trained using the TensorFlow framework. The dataset used contains images of water and soda bottles and cardboard boxes. The strategy used for counting the bottles is based on the identification of the caps, because these are more noticeable, thus optimizing the process. The results of this work show that the system designed facilitates the entry of products through the different functionalities implemented. The training of the object detection network led to an overall accuracy of 90.8%, evaluated by means of a confusion matrix.

Keywords: Artificial vision, Object detection, Python interface, TensorFlow, Grocery stores.

Capítulo I

Introducción

1.1. Planteamiento del problema

El comercio minorista de productos en micromercados y tiendas de víveres es una de las principales actividades económicas del país ya que aproximadamente 520 mil personas viven de esta actividad [1].

Esta actividad carece de sistemas tecnológicos que faciliten el trabajo diario, las personas encargadas de estos negocios deben realizar actividades que pueden llegar a ser repetitivas como por ejemplo el ingreso de los productos del proveedor en el inventario. Es así que el trabajador debe contar manualmente cada producto, lo que puede generar confusión [2].

La visión artificial brinda la posibilidad de reconocer y contar de objetos, con lo cual, se puede implementar un sistema que ayude al trabajador con tareas relacionadas al ingreso de productos enfocados en las botellas y cajas. Además, con la implementación de la facturación electrónica, muchos de estos negocios han adquirido equipos tecnológicos conocidos como sistemas POS cuyas siglas en inglés significan Point Of Sale, que se basan en una computadora que cuenta con una interfaz de venta, que pueden ser usados para este propósito, debido a que el sistema operativo por defecto es Windows.

En base a lo expuesto, implementar un sistema de conteo de botellas y cajas basado en técnicas de visión artificial ayudaría a facilitar el trabajo diario en tiendas de víveres y llevar una mejor organización del inventario.

1.2. Objetivos

1.2.1. Objetivo general

- Desarrollar un sistema de visión artificial para el conteo de cajas y botellas en tiendas de víveres.

1.2.2. Objetivos específicos

- Identificar las condiciones óptimas para la adquisición de imágenes de botellas y cajas.
- Diseñar el algoritmo considerando las condiciones de iluminación y espacio.
- Implementar el sistema diseñado para su posterior validación en condiciones reales de trabajo.

1.3. Justificación

Los sistemas tecnológicos se han convertido en una parte fundamental de nuestras vidas, Por lo cual es importante que nuestras actividades económicas sean facilitadas mediante los avances tecnológicos. Como es sabido a partir del 29/11/2022 en el Ecuador se comienza con la facturación electrónica por lo cual muchos negocios han optado por adquirir equipos computacionales denominados POS [3]. Estos cuentan con características que los hacen factibles para integrar sistemas de visión artificial para automatizar algunos de los aspectos como el manejo del inventario de productos nuevos que son reabastecidos diariamente en las tiendas. El trabajo en las tiendas de víveres tradicionalmente es realizado manualmente por lo cual no presenta un gran desarrollo que le permita ampliar su negocio, mediante el sistema planteado, se pretende mejorar la toma de decisiones ya que al conocer el número de productos que se dispone en el inventario se facilita calcular las futuras compras.

1.4. Alcance

Se pretende realizar un sistema de conteo de botellas y cajas para ser implementado en una tienda de víveres, para facilitar el trabajo de ingresar los nuevos productos que llegan a la tienda diariamente. Este sistema contará con una cámara que será ubicada estratégicamente para que un usuario coloque los productos. Luego mediante una interfaz gráfica se podrán clasificar y contar generando una entrada de datos para ser utilizada en un registro.

Capítulo II

Revisión literaria

En este capítulo, se sintetiza la información utilizada para el desarrollo de la solución propuesta en el capítulo anterior, a través de la revisión de los antecedentes, fundamentos y tecnologías utilizadas. Se abordan los principales aspectos de la implementación de sistemas de visión artificial en la detección y el conteo de productos y cómo ayudan en la automatización de algunas tareas en las tiendas de víveres.

2.1. Antecedentes

En 2018, Tonioni et al., plantean en [4]. El reconocimiento de varios productos presentes en una estantería de una tienda utilizando técnicas de visión artificial, identifican algunos inconvenientes relacionados con la gran cantidad de productos que se pueden encontrar en estos negocios, como solución han encontrado que es posible detectar a los productos similares en la estantería para luego clasificarlos individualmente empleando redes neuronales convolucionales (CNN) y el algoritmo de detección YOLO, el modelo se ha entrenado con una base de imágenes de 1247 elementos, obtenidas desde una cámara ubicada a la altura de un carro de compras. Esto ha permitido que la base de datos tenga variedad y pueda ser usada en varios escenarios. En la solución planteada no se utiliza una fuente de iluminación adicional, el tipo de aprendizaje automático escogido es el supervisado. La principal aplicación de la visión artificial en este trabajo se enfoca en la administración del negocio, destacando que se puede mejorar el manejo de los inventarios.

Bailón y Rodríguez en su trabajo de grado en [5]. Utilizan el framework de TensorFlow y el entorno de programación Python, para crear un sistema que ayuda a clasificar las frutas que se venden en un supermercado. El objetivo principal es automatizar el proceso del ingreso de las frutas en el sistema por parte de los cajeros. Para ello utilizan un ambiente uniforme para evitar el ruido en las imágenes capturadas, que consta de un fondo claro en el mostrador. Han escogido 5 tipos de fruta para crear una base de datos que contiene 100 imágenes por cada clase, para entrenar el modelo de detección basado en una CNN.

Fernández en su trabajo de grado en [6]. Utiliza Einstein Vision de Salesforce, para clasificar los productos de una estantería pequeña en el hogar. Emplea un dispositivo móvil para crear un conjunto de 250 imágenes, que abarcan 9 tipos de productos colocados en una estantería de 3 filas. El objetivo principal del trabajo es detectar sí en alguna fila de la estantería faltaba algún producto, y enviar esa información a un servidor en la nube para poder acceder desde la computadora o un dispositivo móvil. Esto permite que el usuario pueda hacer una lista de compras y reponer el producto. Emplea el tipo de aprendizaje supervisado para entrenar la red neuronal, adicionalmente no utiliza fuente de luz externa, ya que las fotografías son tomadas con luz ambiental.

Moreno y Vera en su trabajo de grado en [7]. Presentan el desarrollo de un sistema de estantería inteligente en el cual una cámara va recorriendo cada fila de la estantería buscando los productos que hacen falta. El objetivo es facilitar y optimizar el proceso de abastecimiento en las tiendas y supermercados, funciona mediante el empleo de un modelo de visión artificial basado en YOLO. Está desarrollado empleando el entorno de programación Python y una base de datos alojada en la nube, además utilizan el tipo de aprendizaje supervisado. Las condiciones de iluminación son adecuadas para permitir que las características de los productos sean notablemente visibles, se utiliza iluminación led.

Moran plantea en [8]. Evaluar el desempeño de algunos modelos de redes neuronales convolucionales desarrollados en TensorFlow, en la clasificación de 10 productos de un supermercado, entre los que se destacan las botellas de aceite. El conjunto de datos para el entrenamiento consta de una muestra de aproximadamente 450 imágenes por cada producto. El entorno de programación elegido es Python, como resultado se evidencia que la

implementación de una red neuronal convolucional es factible para la detección de diversos productos de las tiendas. Adicionalmente no se usa una fuente de luz adicional.

2.2. Tiendas de víveres

Son negocios dedicados al comercio de productos de primera necesidad, por lo general existe al menos una en cada barrio y son conocidas por ofrecer con una amplia variedad de productos y por su cercanía a los hogares [9].

Se ubican dentro de locales en los cuales existen varias estanterías y góndolas usadas para exhibir los productos, agrupándolos por categorías y marcas. Poseen un mostrador principal destinado a la cobranza en donde se encuentra la caja registradora o computadora que es empleada en la emisión de facturas electrónicas (Figura 2.1). La iluminación en esos establecimientos es adecuada, ya que permite una correcta visualización de los productos, siendo las fuentes de iluminación del tipo led o luz natural. Además, tienen una entrada amplia destinada a la atención de los clientes en donde también existe un espacio destinado al recibimiento de la mercancía. El horario de atención de estos negocios es de 6:00 am a 11:00 pm.

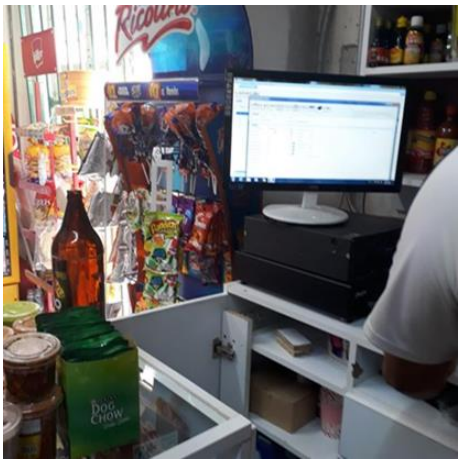

Figura 2.1: Tienda de víveres.

La mayoría de los productos comercializados dentro de las tiendas de víveres se transportan en cajas de cartón o se envasan en botellas de plástico. Son abastecidos periódicamente, ya sea cada semana o cada 15 días.

2.2.1. Botellas

La mayoría de los productos líquidos se envasan en botellas de plástico, esto permite abaratar el costo del recipiente. Se fabrican en serie mediante el proceso de soplado de plástico PET. Estas botellas, se encuentran en distintas formas y tamaños y siendo utilizadas en productos tales como: gaseosas, agua, aceite y productos de limpieza [10].

A nivel físico, la mayoría de las botellas carecen de color, ya que son transparentes y permiten mostrar el contenido en su interior. Las tapas se encuentran en distintos colores y tienen una forma similar en la mayoría de las botellas, independientemente del uso, tienen un diámetro aproximado de 30mm y una altura que ronda los 15mm. La similitud de las tapas, principalmente, se debe a que las botellas se fabrican a partir de una preforma y se utilizan diferentes moldes para dar forma al cuerpo de la botella, dependiendo del uso. En el caso de las bebidas gaseosas y el agua, tienen diversos tamaños con presentaciones de 340ml, 1l, 3l, como se ve en la Figura 2.2.

Figura 2.2: Preforma de botellas plásticas y ejemplos [11].

2.2.2. Cajas de cartón

Son utilizadas ampliamente para el embalaje y el transporte de otras mercancías más pequeñas, como es el caso de los productos líquidos enfundados o enlatados galletas, confitería, jabones, etc. [10].

Estas cajas están fabricadas de cartón corrugado y se encuentran en diversos tamaños, siendo las dimensiones promedio de 30cm x 40cm x 30cm. La cara más amplia se usa para mostrar la mayor cantidad de información, en la cual se incluyen logotipos, cantidad y nombre del producto. En la cara angosta y en la parte superior se encuentran indicaciones sobre el

apilamiento correcto de las cajas, señalética sobre la manipulación y el transporte, información adicional e identificadores del producto como códigos de barras (Figura 2.3).

Figura 2.3: Caja de cartón en la que se transporta aceite.

2.3. Visión artificial

La visión artificial es una rama dentro de la inteligencia artificial, en la cual el principal objetivo es permitir que el computador o máquina, sea capaz de percibir e interpretar el entorno visual, por medio del análisis de imágenes o videos. De cierta manera, se imita el proceso natural en que los humanos reconocemos el mundo que nos rodea a través de la vista y el cerebro. Esto se logra a través del desarrollo de dispositivos de captura de imágenes (vista), y en procesamiento de los datos mediante algoritmos (cerebro). Un sistema básico de visión artificial está conformado de las siguientes etapas: adquisición de imagen, preprocesamiento de imagen y análisis de imágenes [12].

El inicio de esta disciplina se da en la década de 1950, en donde se desarrolla el sistema Mark I Perceptrón para el análisis de imágenes, inicialmente destinado para formar parte de un proyecto en el que se buscaba la interpretación de fotografías de satélites empleando técnicas de análisis computacional [13]. Luego, en la década de 1960, se plantean nuevos modelos matemáticos enfocados en recrear el proceso de la visión, a partir de las investigaciones realizadas en mamíferos (felinos y primates). En las cuales se observa que los bordes de los objetos dentro de las imágenes producen una respuesta más notoria en la corteza cerebral [14].

Un algoritmo para la identificación de los bordes de los objetos en una imagen es el desarrollado por Sobel en 1968, en el cual se usan *kernels* (filtros) de convolución verticales y horizontales, de tamaño 3x3, que recorren la imagen [15]. Esto se muestra en la Figura 2.4.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Figura 2.4: Filtro horizontal y vertical [15].

Posteriormente, en las siguientes décadas, se dieron grandes avances al desarrollar nuevos algoritmos de extracción bordes como Canny y Prewitt, entre otros. Las redes neuronales convolucionales, han permitido mejorar notoriamente el desempeño en la detección y reconocimiento de objetos [15], [16].

En la Figura 2.5, se puede observar la aplicación de los algoritmos de detección de bordes.

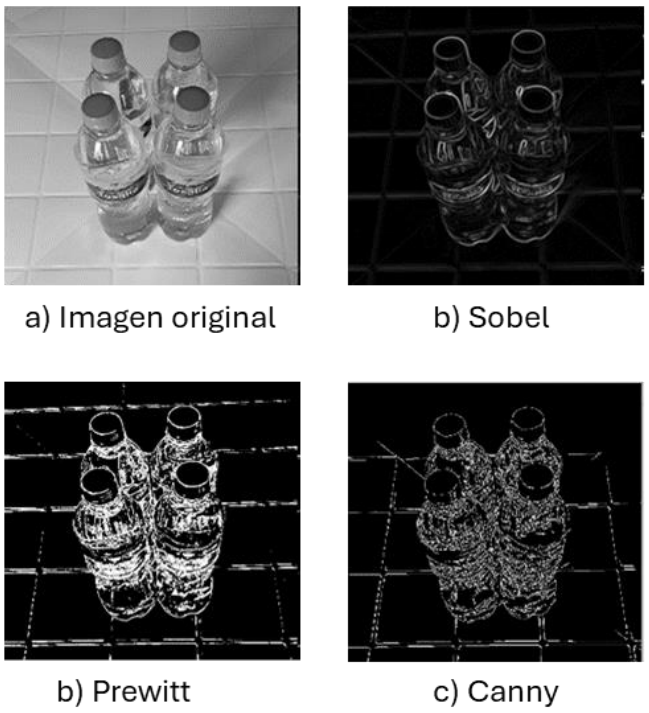

Figura 2.5: Algoritmos de detección de bordes.

La visión artificial se ha empleado ampliamente en múltiples campos. Se destaca principalmente en la detección de enfermedades de plantas en la agricultura, facilitando el diagnostico radiográfico en el área de medicina, contribuyendo en los sistemas de control de calidad en el sector industrial y hoy en es una parte fundamental en la conducción autónoma de vehículos.

2.3.1. Adquisición de imágenes

En esta etapa, se obtienen imágenes digitales 2D que representan el entorno visual. Las imágenes son un conjunto de valores binarios organizados en una cuadrícula. Esto se realiza mediante un proceso de conversión análoga-digital utilizando un dispositivo de captura.

2.3.1.1. Dispositivos de captura

Comúnmente, a estos dispositivos se los conocen con los nombres de cámaras o escáneres. Son aparatos eléctricos que cuentan con un sensor fotosensible, que está dividido en celdas, que se denominan píxeles. Cada píxel está compuesto por materiales semiconductores que generan señales eléctricas cuando reciben la luz reflejada de los objetos en el entorno [17].

Los sensores están diseñados para capturar la mayoría del espectro visible de la luz. Estos cuentan con diversos lentes y filtros ópticos que concentran la luz y bloquean el espectro no visible como el infrarrojo y el ultravioleta. La cantidad de píxeles del sensor determina la resolución del dispositivo, esta cifra se expresa en mayormente en megapíxeles. La relación de aspecto se obtiene al dividir la cantidad de píxeles de ancho sobre la cantidad de píxeles de alto, siendo las siguientes las relaciones de aspecto más comunes: 1:1, 4:3, 3:2 y 16:9, que se muestran en la Figura 2.6.

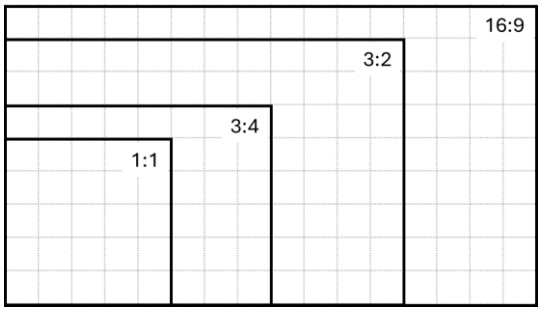

Figura 2.6: Formatos en la adquisición de video.

Las imágenes a color se obtienen mediante tres subpíxeles encargados de la detección de un color específico (rojo, verde y azul). Cada color se lo conoce como canal, al combinar el valor de cada uno de estos, se pueden representar la mayor parte del espectro de luz visible. Esto es similar a la estructura biológica de los ojos humanos, en donde existen 3 tipos de células fotorreceptoras llamadas conos para cada color [18].

2.3.1.2. Iluminación en la visión artificial

La iluminación es necesaria para lograr resaltar las características de los objetos a detectar. Existen diversas fuentes de luz tales como: solar, led, incandescente, fluorescente y halógena. La diferencia entre estas radica en la potencia con que emiten las diversas longitudes dentro del espectro de luz visible, como se ve en la Figura 2.7. Cada tipo de iluminación presenta picos de emisión en determinados colores. La elección de la fuente de iluminación se basa en asegurarse que todos los colores de los objetos a detectar están dentro del espectro de emisión y se puedan observar de manera correcta [19].

En este caso, la luz natural tiene una mejor distribución en el espectro en comparación a las demás. Las fuentes incandescentes y halógenas mayormente emiten en el rango infrarrojo. Las fuentes fluorescentes, presentan algunos picos en la región ultravioleta. En el caso de la iluminación led existen de 2 tipos, la cálida y fría que mayormente se utilizan combinadas para mejorar el espectro de emisión.

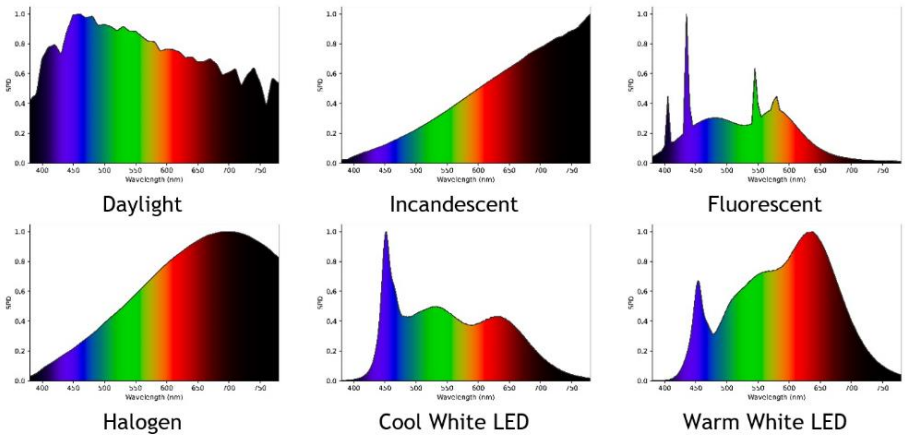

Figura 2.7: Espectro de emisión de algunas fuentes de luz [19].

Otras consideraciones importantes, son la eficiencia energética y la ubicación de la fuente de luz. La ubicación es esencial, ya que se puede generar sombras o reflejos en los objetos. Además, dependiendo de la intensidad de la fuente y la distancia a los objetos se puede saturar el sensor.

2.3.2. Preprocesamiento de imágenes

El preprocesamiento de la imagen consiste en la corrección de los posibles defectos que puede tener originalmente. Principalmente en esta etapa se busca en reducir el ruido, mejorar

el brillo o el color, realizar recortes, modificar el tamaño, cambiar a escalas de grises o otras correcciones.

2.3.2.1. Espacios de color

Existen algunos espacios de colores como RGB y CIELAB (Figura 2.8). Se diferencian en la manera que son representados los colores. En el espacio RGB se utilizan tres canales (Red, Green, Blue), cualquier color se representa mediante es la combinación, del valor de la intensidad de cada canal, estos valores se almacenan con números enteros de 8 bits de 0 a 255, por lo cual existirían 256^3 colores (16'777.216). El espacio de color CIELAB se basa en una distribución que toma en cuenta la percepción humana del color, utiliza tres canales L*a*b, (Luminosidad, (verde, rojo), (azul, amarillo)), los valores son flotantes y los rangos de los canales son: "L" entre [0, 100], "a" entre [-128, 128] y "b" entre [-128, 128], este espacio de color permite comparar la diferencia entre colores de una manera más exacta [20].

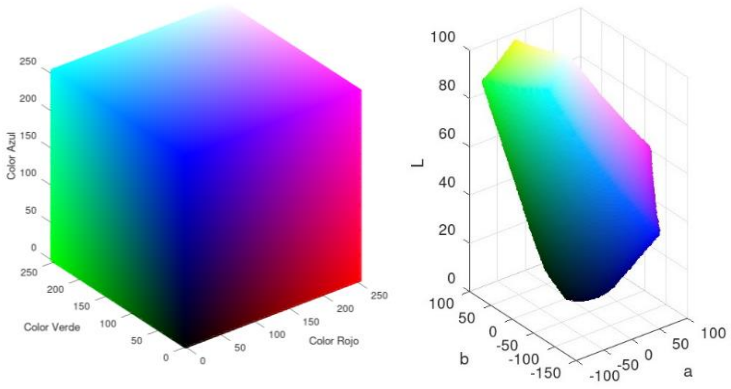

Figura 2.8: Espacios de color RGB y CIELAB.

2.3.2.2. Corrección de imágenes

Las imágenes pueden ser modificadas con el fin de obtener una mejor representación de los detalles de los objetos, es decir que sean más notables. Las operaciones más básicas consisten en el redimensionado, recortes y rotaciones. La corrección de imperfecciones se basa en ajustes de contraste y brillo, ecualización de canales, corrección de color, eliminación de ruido, suavizado de bordes, colocación de filtros y máscaras, etc.

En una imagen con contraste bajo, la ecualización de cada canal permite la visualización de detalles aparentemente ocultos Figura 2.9.

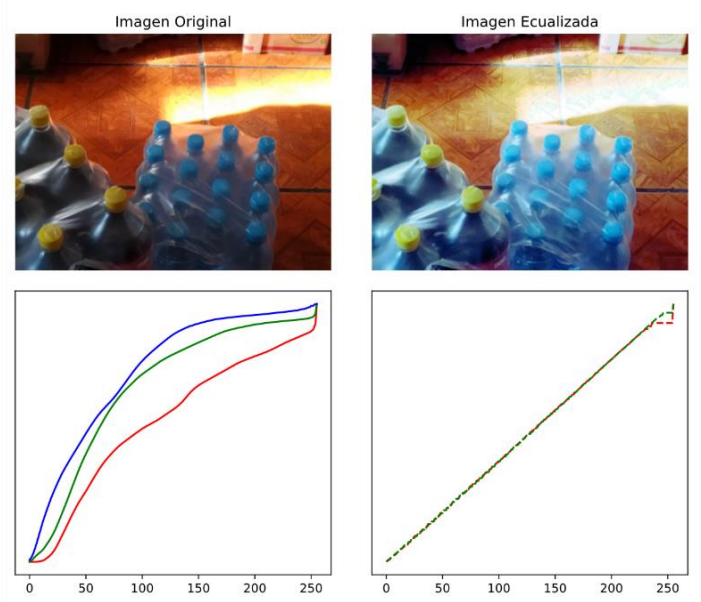

Figura 2.9: Ecualización de una imagen e histograma de color.

2.3.3. Análisis de imágenes

Esta etapa consiste en la detección e identificación de los objetos presentes en una imagen, por medio de estrategias y algoritmos de análisis computacional.

2.3.3.1. Redes Neuronales

Las redes neuronales son el pilar dentro del aprendizaje automático en la computación, ya que permiten la resolución de tareas en diversos ámbitos. Se componen de una entrada y una salida, entre estas se encuentran numerosas neuronas interconectadas, organizadas en capas, el funcionamiento es similar a las conexiones nerviosas del cerebro humano y siguen el esquema de la Figura 2.10 [21].

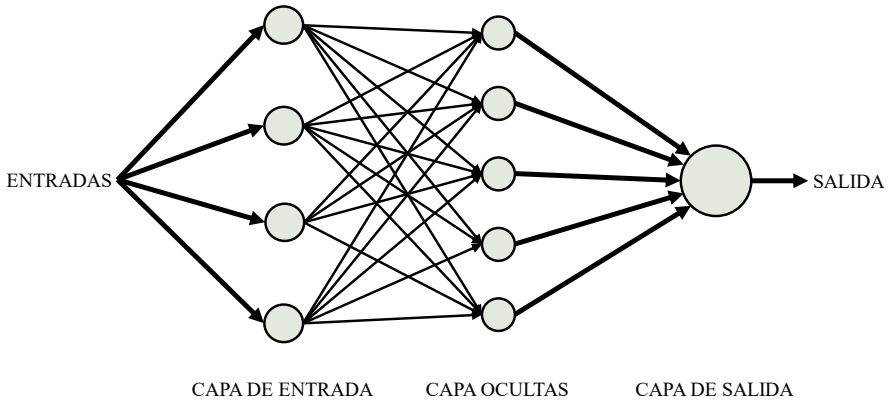

Figura 2.10: Esquema básico de una red neuronal artificial [21].

Cada neurona tiene la capacidad de activarse o desactivarse en función de los valores que recibe en su entrada. Cada entrada se asocia con un valor denominado *peso*, estos se ajustan durante en el entrenamiento y cambian hasta que se minimiza el error entre la salida de la red neuronal y el resultado esperado. En el aprendizaje automático de las redes neuronales, es necesario proporcionar suficientes datos de entrada y salida para obtener un buen desempeño. El tipo de redes neuronales más avanzadas son las convolucionales por su arquitectura y son utilizadas ampliamente en la inteligencia artificial debido a su versatilidad de adaptación en diversos escenarios en el análisis de imágenes, al incluir capas de convolución que aplican filtros o "kernels" [22].

2.3.3.2. Aprendizaje supervisado

Es un tipo de aprendizaje automático en el cual el modelo se entrena utilizando una base de datos de salida previamente etiquetada, estos datos son usados después en el entrenamiento del modelo, en el cálculo del error entre la salida de la red neuronal y los resultados esperados, en función a este cálculo se ajustan los valores de los pesos [23].

En el caso de la detección de objetos en visión artificial, los datos etiquetados corresponden a ubicación de los objetos dentro de cada imagen. Mayormente se utiliza un recuadro para indicar la presencia de determinado objeto. Este recuadro se grafica usando dos puntos de valores mínimos y máximos en el eje X y el eje Y. Como se observa en la Figura 2.11, el punto P1 es (X_{min}, Y_{min}) y P2 es (X_{max}, Y_{max}), cabe mencionar que el origen de las coordenadas de una imagen se sitúa en la parte superior del lado izquierdo.

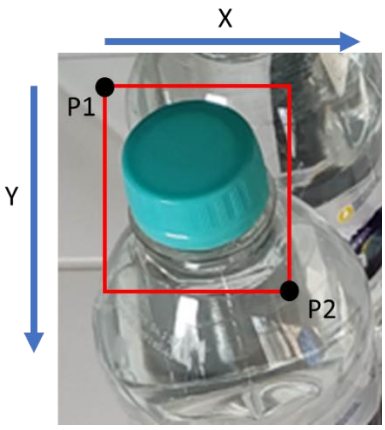

Figura 2.11: Ubicación de las coordenadas de un objeto.

2.3.3.3. Modelo SSD MobileNet V2 FPNLite 320x320

Es un modelo de detección de objetos desarrollado por Google, enfocado en la reducción del costo computacional. La entrada de este modelo es una imagen a color de 320 x 320 píxeles y la salida es un tensor que contiene un índice de identificación, la confianza y las coordenadas de la ubicación del objeto detectado. Adicionalmente, está preentrenado con la base de datos de MS COCO de alrededor de 320 mil imágenes etiquetadas en 90 clases [24].

Integra las siguientes capas:

- Red convolucional MobilNetV2
- Extractor de características Feature Pyramid Network (FPN)
- Red de detección Single Shot Detector (SSD)

Cada capa cumple una tarea específica, la capa de Mobilnet V2 es una red neuronal convolucional de 53 capas que está optimizada para la extracción de las características de los objetos presentes en una imagen, permitiendo una rápida respuesta en dispositivos con recursos limitados de procesamiento [25]. Los algoritmos empleados se enfocan en simplificar el número de operaciones realizadas. Para ello se ocupa la "separación de convoluciones en profundidad" en la cual se aplica una convolución para filtrar ligeramente cada canal de entrada, para luego aplicar una capa de convolución puntual que calcula las combinaciones lineales de cada canal. El costo computacional se ve reducido de 8 a 9 veces al momento de aplicar el filtro o *kernel* [24].

La capa Feature Pyramid Network también es una red neuronal convolucional que se encarga de la creación de mapas de características de los objetos en distintas escalas en una imagen. Su característica principal es la rapidez, como entrada utiliza la salida de un modelo de extracción de características como MobileNetV2 [26].

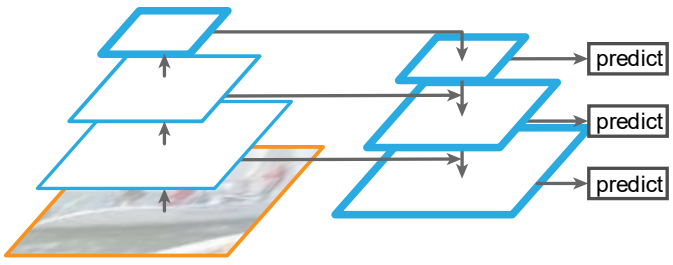

Figura 2.12: Detección de objetos a múltiples escalas FPN [26].

La capa Single Shot Detector es una red neuronal convolucional que permite la detección de múltiples objetos en una imagen en un solo paso. Es la capa final ya que toma los resultados de las capas anteriores e identifica los objetos al dividir a la imagen en cuadriculas y se examina en cada una de estas si existe la presencia de un objeto [27].

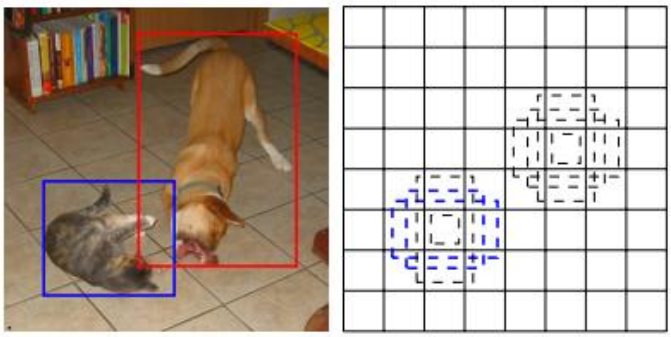

Figura 2.13: Estrategia de detección del modelo SSD [27].

2.3.3.4. El dataset

Consiste en la creación de una base de datos que es utilizada en el entrenamiento del modelo de detección. Contiene cientos de imágenes y archivos que en los que constan la ubicación precisa de los objetos presentes dentro de cada imagen y un identificador o "clase". La creación del dataset se realiza de manera manual, creando imágenes con los dispositivos de captura y los archivos de las etiquetas empleando diversas herramientas informáticas.

MAKEAISENSE es una página web que es usada en la creación de datasets para la detección de objetos, facilita la tarea del etiquetado de las imágenes y permite la exportación de archivos en diversos formatos. Tiene una interfaz intuitiva y fácil de usar como se muestra en la Figura 2.14

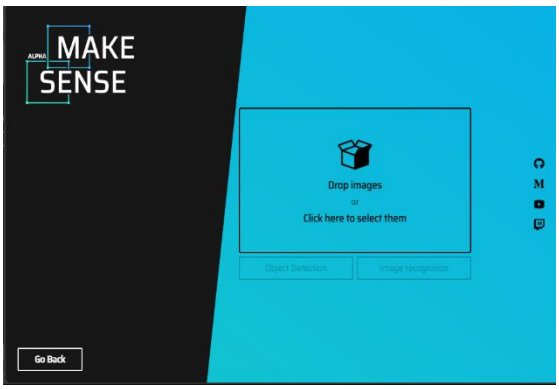

Figura 2.14: Pantalla de carga de Make Sense AI.

El formato en el cual se almacenan las coordenadas del recuadro de cada objeto en una imagen puede ser los siguientes:

- **Formato YOLO:** Este formato, las coordenadas del recuadro se almacenan en un archivo txt, mediante 5 valores: (Id, X_{centro} , Y_{centro} , Ancho, Alto). Estos valores se encuentran normalizados a excepción del Id y son proporcionales a las dimensiones de la imagen, esto permite que la imagen pueda ser redimensionada sin afectar a la ubicación de los recuadros.
- **Formato XML** Es un formato que se organiza en una estructura jerarquizada, en el cual consta la información de las dimensiones y ruta de la imagen, nombre del objeto, coordenadas en píxeles. Este tipo de formato permite la búsqueda de atributos con mayor facilidad.

Es importante que exista una considerable cantidad de imágenes que muestren a los objetos en una amplia variedad de escenarios. Posteriormente la cantidad de imágenes será dividida en dos conjuntos: 80% para el entrenamiento y el 20% para la validación.

2.3.3.5. Entrenamiento del modelo

Consiste en el ajuste de los pesos del modelo, esto se consigue al ingresar las imágenes del dataset para luego comparar los resultados de la detección de objetos, frente a los que se encuentran etiquetados (salidas esperadas frente a datos de entrada conocidos).

Se trata de un proceso iterativo en el cual se configuran algunos parámetros tales como: el número de veces que se realiza (pasos o épocas), la cantidad de imágenes que se van a introducir en cada iteración, los puntos de control, entre otros. En este proceso se consumen muchos recursos de procesamiento, ya que se realizan numerosos cálculos.

Para disminuir el tiempo empleado, se pueden utilizar tarjetas GPU (Unidad de Procesamiento Gráfico) o TPU (Unidad de Procesamiento Tensorial). Existen algunos servicios en la nube que permiten el uso de estas tarjetas, el que más se destaca es Google Colab ya que se puede utilizar gratuitamente por un periodo de tiempo considerable. (Figura 2.15.)

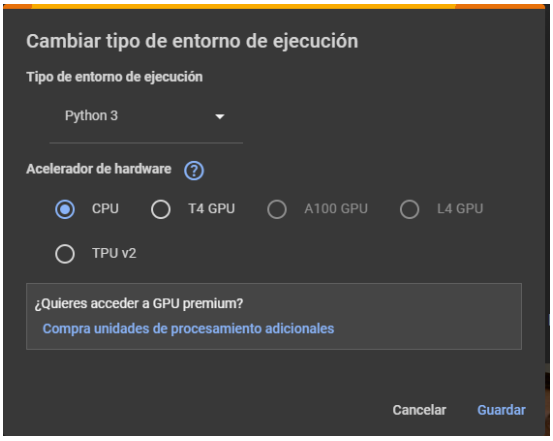

Figura 2.15: Tarjetas de procesamiento disponibles en Google Colab.

2.3.3.6. Comparación de color

El análisis del color se utiliza para identificación de características u objetos en una imagen [28]. Para comparar dos colores, se puede calcular la distancia euclidiana entre ellos, dependiendo del espacio de color utilizado el resultado puede ser más o menos exacto, la fórmula para calcular la distancia entre 2 puntos en 3 dimensiones es:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{2.1}$$

El espacio de color CIELAB permite comparar colores de mejor manera, al calcular la distancia en los colores expresados en coordenadas L, a, b se obtiene un valor que representa de mejor manera la diferencia entre los colores y es conocido como Delta E ($\Delta E*$). En la Tabla 2.1, se muestra una comparativa de estos valores.

Tabla 2.1: Diferencia de color Delta E.

Distancia	Observación
<=1	Diferencia no percibida por el ojo humano
1-2	Diferencia percibida al acercase
2-10	Diferencia visible a distancia
11-49	Los colores son similares.

2.4. Herramientas de desarrollo

2.4.1. Python

Python es un lenguaje de programación de alto nivel de código abierto que destaca por tener una sintaxis intuitiva. Se usa en una amplia gama de aplicaciones y es el más usado en el aprendizaje automático. Permite la programación orientada a objetos y además posee una extensa comunidad que desarrolla múltiples librerías que facilitan la programación. En su instalación se incluye un editor de código llamado IDLE que facilita la programación.

Las librerías que se emplean a menudo son:

- **OpenCV:** Es una librería con múltiples herramientas para visión por computadora, se utiliza en la captura, procesamiento y análisis de imágenes y videos.
- **NumPy:** Se usa ampliamente en cálculos matemáticos, operaciones con matrices y vectores.
- **Tkinter**: Es una librería utilizada en la creación de interfaces gráficas de usuario, permite la creación de ventanas, con múltiples tipos de componentes que facilitan el uso del programa por parte del usuario.
- **Matplotlib**: Contiene múltiples funcionalidades que permiten la representación de datos en gráficas 2D y 3D.
- **OS**: Es una librería que permite interactuar con el sistema operativo, permite crear, eliminar y consultar directorios, configurar variables de entorno, ejecutar comandos, obtener información del sistema, etc.

2.4.2. TensorFlow

Es una *framework* de código abierto desarrollado por Google, para la investigación y el desarrollo de tecnologías de inteligencia artificial. Las funciones principales son la creación y el entrenamiento de modelos basados en redes neuronales, permite el despliegue de aplicaciones tanto en computadoras (CPU / GPU), dispositivos móviles y microprocesadores.

2.4.3. Bash

Es un lenguaje de programación básico utilizado en la consola de múltiples sistemas operativos. En Python se lo utiliza para para la instalación de librerías y creación de entornos virtuales. Además, permite la creación de programas que pueden modificar el sistema, estos se

conocen como “Archivos por lotes” y tienen una extension .bat, se pueden modificar utilizando el Bloc de notas, ya que son archivos de texto.

2.4.4. Octave

Es un lenguaje de programación de código abierto que forma parte de GNU (Proyecto de creación de software libre). Es ampliamente usado en áreas de ingeniería y ciencia para la realización de cálculos numéricos, cuenta con diversas funciones predefinidas que facilitan la programación y la visualización de datos mediante gráficas 2D y 3D (Figura 2.16), que pueden ser configuradas a voluntad, se pueden realizar múltiples operaciones con matrices y números complejos, resolución de ecuaciones, comunicación serial, visión artificial, entre otras funciones. Además, los scripts son compatibles con otros programas como Matlab.

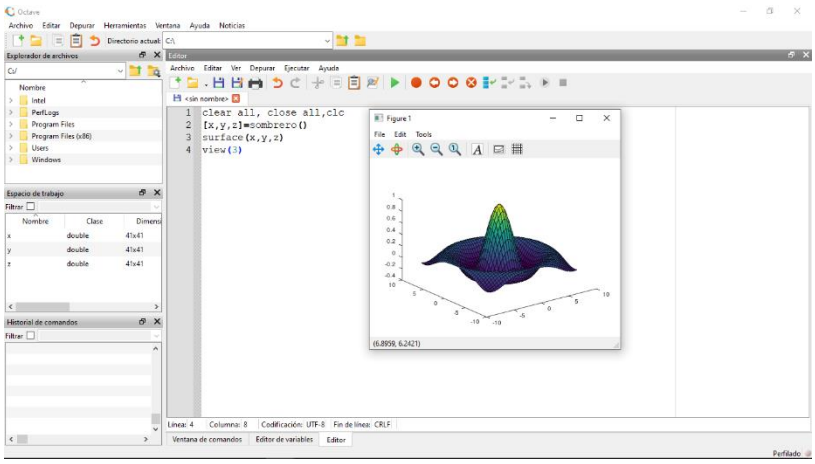

Figura 2.16: Interfaz de Octave.

2.5. Conclusión del estado del arte

En base a los antecedentes y tecnologías descritas anteriormente, se evidencia la factibilidad del desarrollo de un “Sistema de conteo de cajas de cartón y botellas basado en visión artificial para tiendas de víveres”, utilizando diversas estrategias de detección y entornos de desarrollo.

En cuanto a la detección de productos, se destaca principalmente el uso de redes neuronales convolucionales. Principalmente se debe a la capacidad de adaptación que tienen para la detección de objetos, en diversos escenarios y condiciones. Además, se observa la existencia de modelos de detección de objetos que emplean diferentes tipos de algoritmos para reducir el costo computacional y ampliar la operabilidad en diversos dispositivos.

Asimismo, se ha observado que en algunos casos no era necesario una fuente de iluminación adicional, siendo la utilizada la propia del establecimiento o la luz natural.

Por lo tanto, se concluye que el sistema a diseñar consta de dos componentes principales: el modelo de detección y la interfaz gráfica del usuario. El sistema propuesto está enfocado a adaptarse a los recursos tecnológicos una tienda de víveres. Por lo cual se plantea un desarrollo basado en software de código abierto.

Modelo de detección:

- Lenguaje de programación: Python.
- Entorno de desarrollo: Google Colab.
- Framework: TensorFlow.
- Modelo: SSD MobileNet V2 FPNLite 320x320.

Interfaz gráfica del usuario:

- Lenguaje de programación: Python (interfaz) y Bash (iniciador del sistema).
- Editor de código: IDLE (Python 3.9 64-bit) y Bloc de notas de Windows (para Bash).

Capítulo III

Desarrollo

En este capítulo, se detalla el proceso del desarrollo del sistema de conteo de botellas y cajas de cartón. Se plantea un algoritmo en base al cual se crean los diagramas de flujo que describen el funcionamiento del sistema, considerando los requerimientos.

3.1. Diseño del algoritmo

En base a la revisión literaria, se desarrolla un algoritmo representado en la Figura 3.1, el cual describe los pasos necesarios para el desarrollo del sistema de visión artificial. Dentro del primer paso se consideran los aspectos de iluminación y espacio, es decir, las imágenes obtenidas deben tener variedad para que el modelo tenga la capacidad de detectar objetos al existir variaciones en la iluminación y en diversas posiciones.

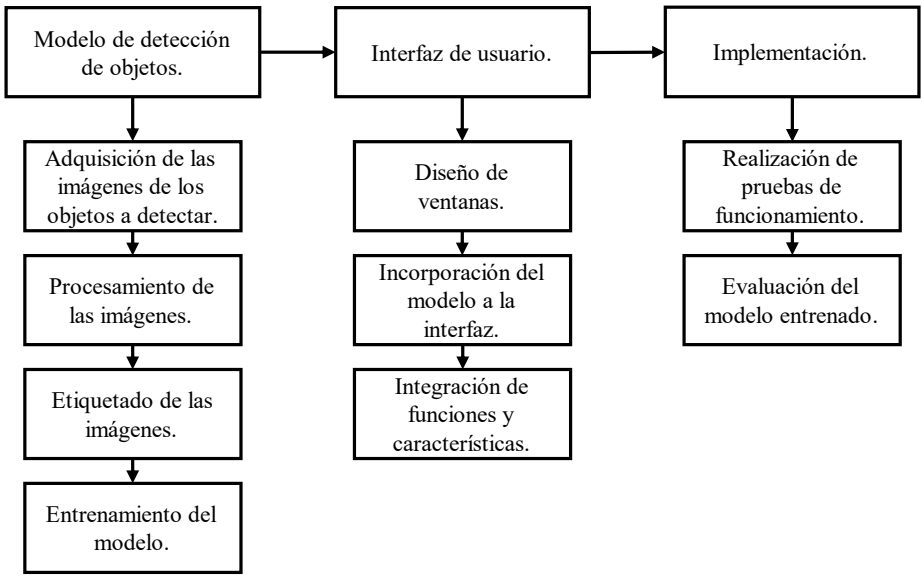

Figura 3.1: Algoritmo diseñado.

3.2. Metodología

Para este proyecto, se utiliza el modelo “cascada”, que organiza las fases del desarrollo de actividades de manera secuencial, es decir, cada fase debe ser completada para avanzar a la siguiente, esto genera una mejor planificación y permite un mejor aprovechamiento del tiempo. Las fases se dividen en:

Requisitos básicos: Se relacionan directamente con el cumplimiento de los objetivos, y describen las funciones mínimas que el sistema debe contar.

Diseño del sistema: Se realizan una serie de actividades con el fin de desarrollar el sistema, considerando los requisitos de la fase anterior, incluye el modelo de detección de objetos y la interfaz.

Implementación: Consiste en evaluar el desempeño luego de la instalación del sistema en condiciones reales de trabajo, en esta fase se recopilan datos para su posterior análisis.

3.3. Estructura del sistema de detección y conteo

El sistema diseñado se presenta en la Figura 3.2, a nivel de hardware se utiliza una computadora y una cámara externa. La interfaz de usuario se encarga de facilitar el uso del modelo de detección de objetos.

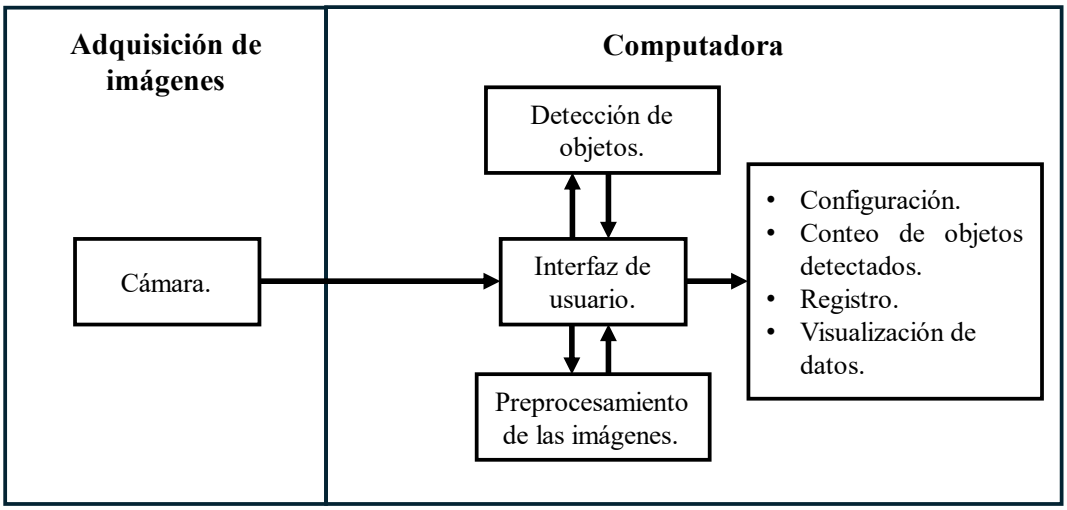

Figura 3.2: Estructura del sistema.

3.4. Requerimientos del sistema

En base al contexto de una tienda de víveres que se ha presentado en la revisión literaria, se identificaron los siguientes aspectos principales:

- La computadora utilizada para la emisión de facturas electrónicas carece de una tarjeta gráfica y un procesador de alto rendimiento.
- Se cuenta con una pantalla amplia para mostrar el contenido.
- Los usuarios prefieren interactuar con los programas utilizando botones, especialmente, cuando se cuenta con dispositivos de cómputo con entrada táctil.
- Para el ingreso de texto se cuenta con un teclado externo.

Al considerar los aspectos anteriormente mencionados, se determinan los siguientes requerimientos básicos para el sistema:

- El sistema debe tener un buen desempeño en equipos de cómputo con procesamiento limitado.
- La interfaz de usuario debe de ser de fácil operación, agilitando la realización de las actividades por medio de botones.
- La distribución de los componentes de la interfaz debe estar optimizada para garantizar su visualización adecuada en la pantalla.

Además, se consideran los siguientes requerimientos que mejoran la experiencia de usuario:

- La instalación del sistema debe de ser fácil.
- Los registros deben almacenarse en formatos conocidos y accesibles desde otros programas.
- El sistema debe mostrar el ingreso de los productos por períodos de tiempo.

3.5. Diseño del modelo de detección de objetos

El modelo de detección de objetos se ha desarrollado en dos fases. La primera fase contempla la creación de la base de datos de imágenes y etiquetas, abarca desde la selección del dispositivo hasta la creación del archivo zip que contiene los datos. En la segunda fase se

obtiene el modelo listo para la detección de los objetos, inicia con la configuración del entorno de entrenamiento y finaliza con la conversión a un formato utilizable.

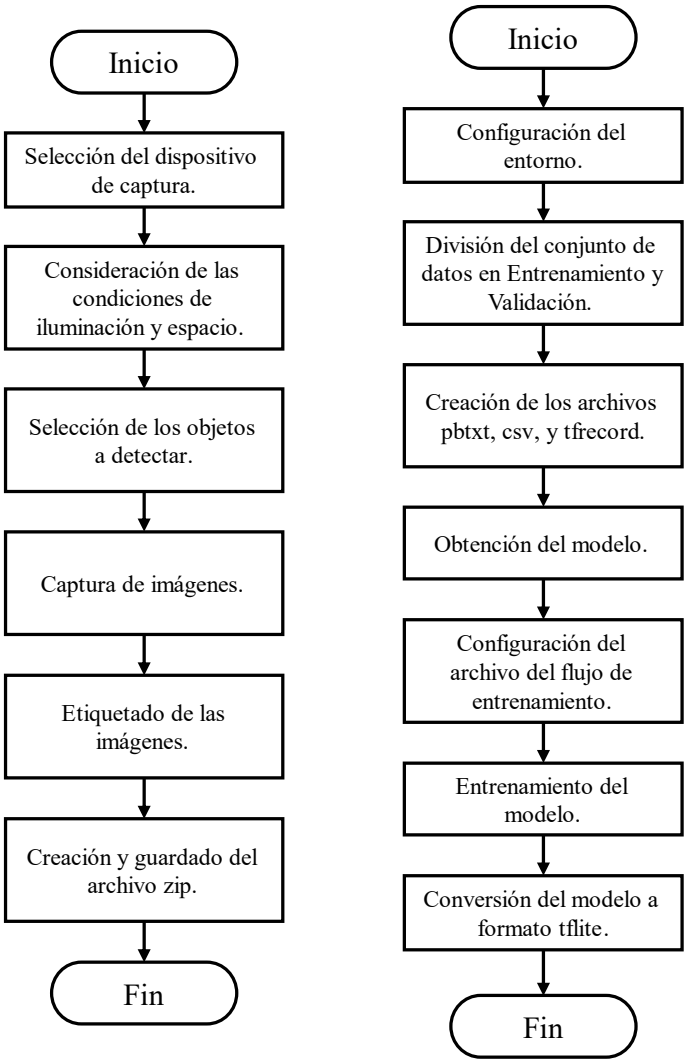

Figura 3.3: Diagramas de flujo del dataset y el entrenamiento del modelo.

3.6. Diseño de la interfaz de usuario

Para cumplir el requerimiento de fácil instalación se plantea en el diagrama de flujo mostrado en la Figura 3.4 la utilización de un entorno virtual de Python en el cual, se instala todas las librerías necesarias, mediante esto se evita problemas debido a conflictos entre versiones de librerías.

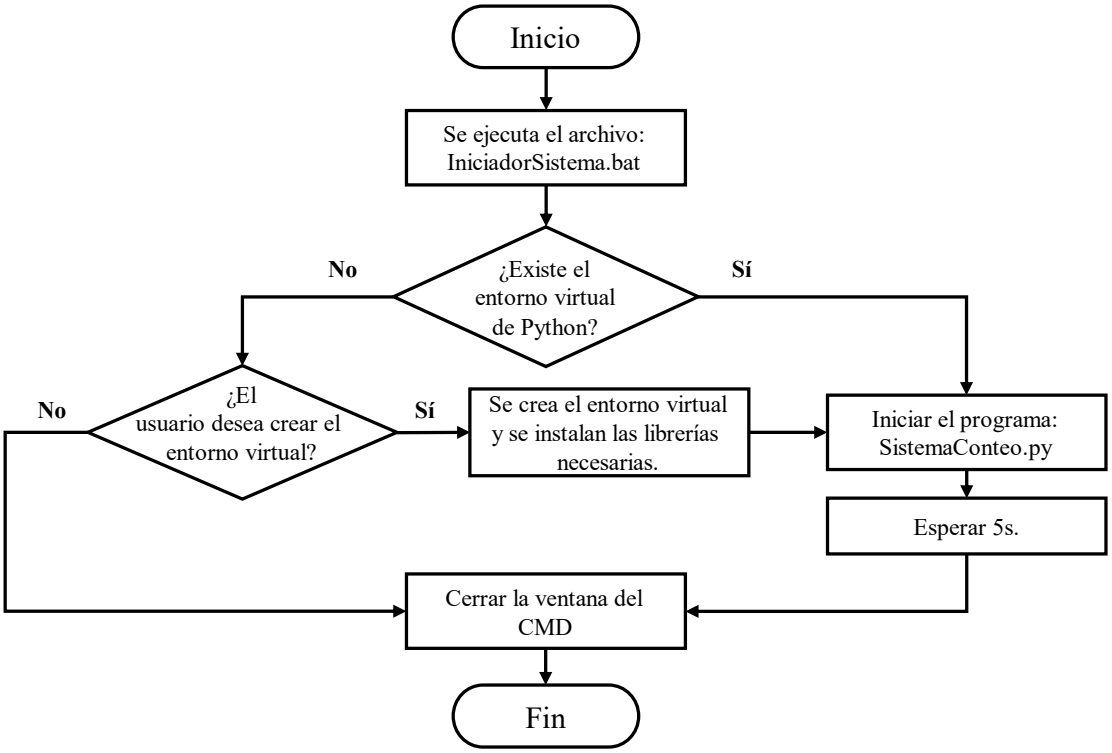

Figura 3.4: Diagrama de flujo del iniciador del sistema.

La interfaz de usuario cuenta con las siguientes ventanas:

- Ventana principal
- Ventana de configuración
- Ventana de visualización del registro

Dentro de la ventana principal se integran múltiples funciones que permiten la captura, preprocesamiento y análisis de las imágenes provenientes del dispositivo de captura, al ejecutar el archivo se sigue el diagrama de flujo de la Figura 3.5.

Además, se realiza la verificación de la configuración en la cual se revisa que en el directorio en donde se encuentra el archivo de la interfaz existan los archivos de **configuración**, **registro.csv** y **registroconteo.csv**. En caso de no existir alguno de estos, se crean automáticamente. La función de los botones en la ventana principal, esta implementada en base al diagrama mostrado en la Figura 3.8.

La ventana de configuración se basa en el diagrama mostrado en la Figura 3.6, en esta se pueden configurar algunos parámetros principales, origen del video, ruta del modelo TFlite, ruta de las etiquetas, umbral de detección y el tiempo en que se muestran los recuadros sobre la imagen.

La ventana de visualización de los registros se programa en base al diagrama mostrado en la Figura 3.7. Para facilitar la comparación de productos ingresados se presentan en base a los periodos de reposición de productos mencionados en el capítulo II.

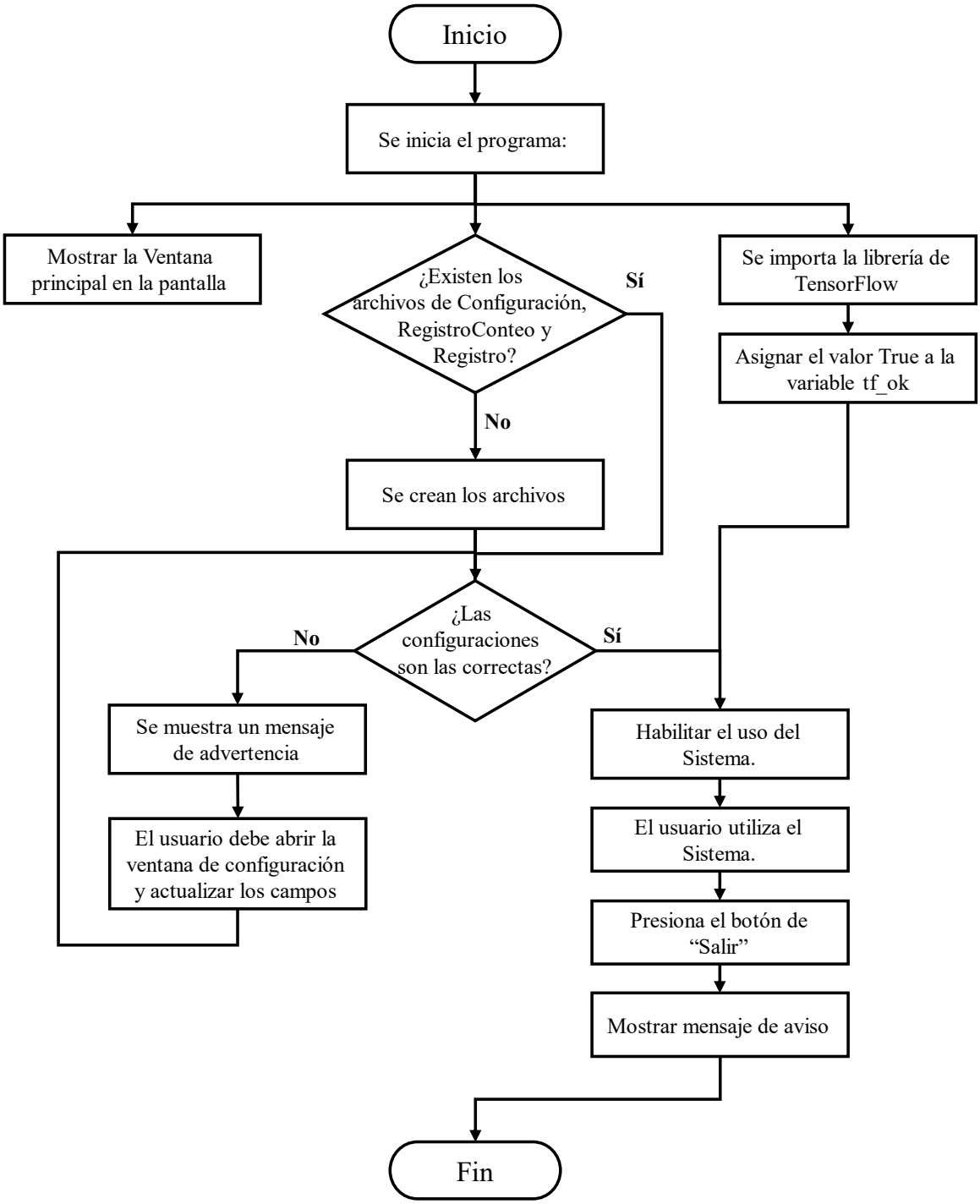

Figura 3.5: Diagrama de flujo de inicio de la interfaz de usuario.

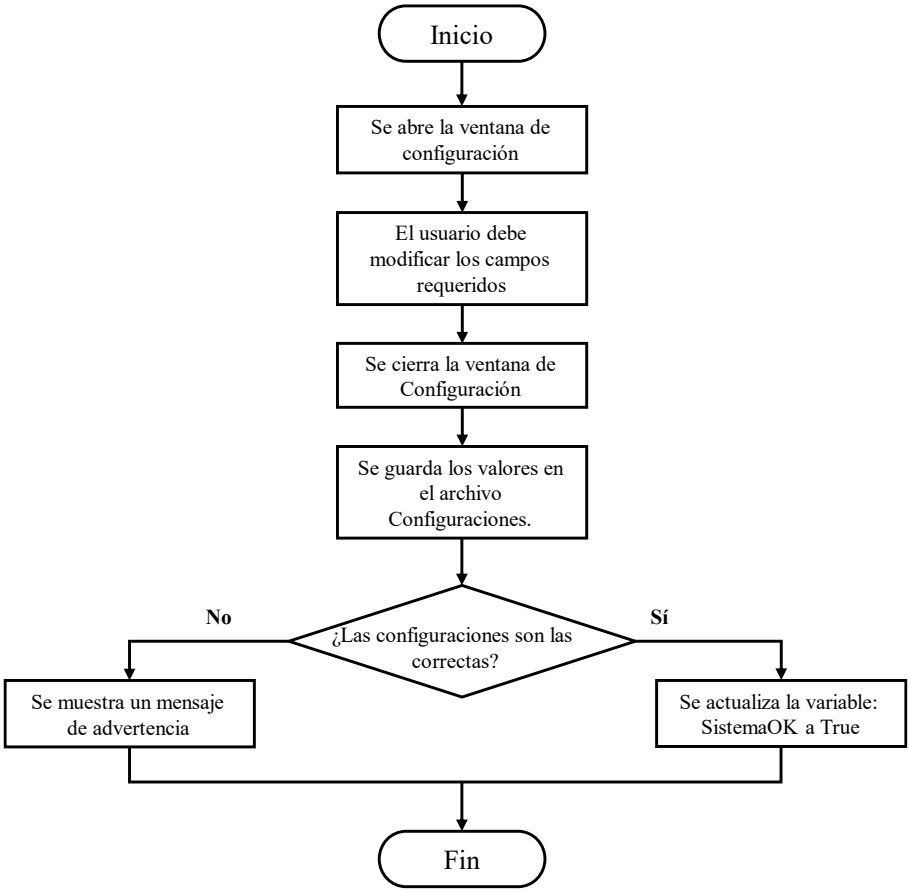

Figura 3.6: Diagrama de flujo de la ventana de configuración.

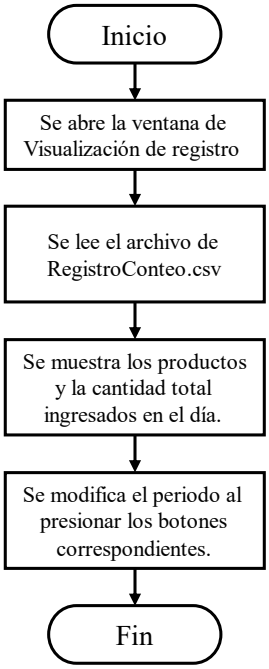

Figura 3.7: Diagrama de flujo de la ventana de visualización del registro.

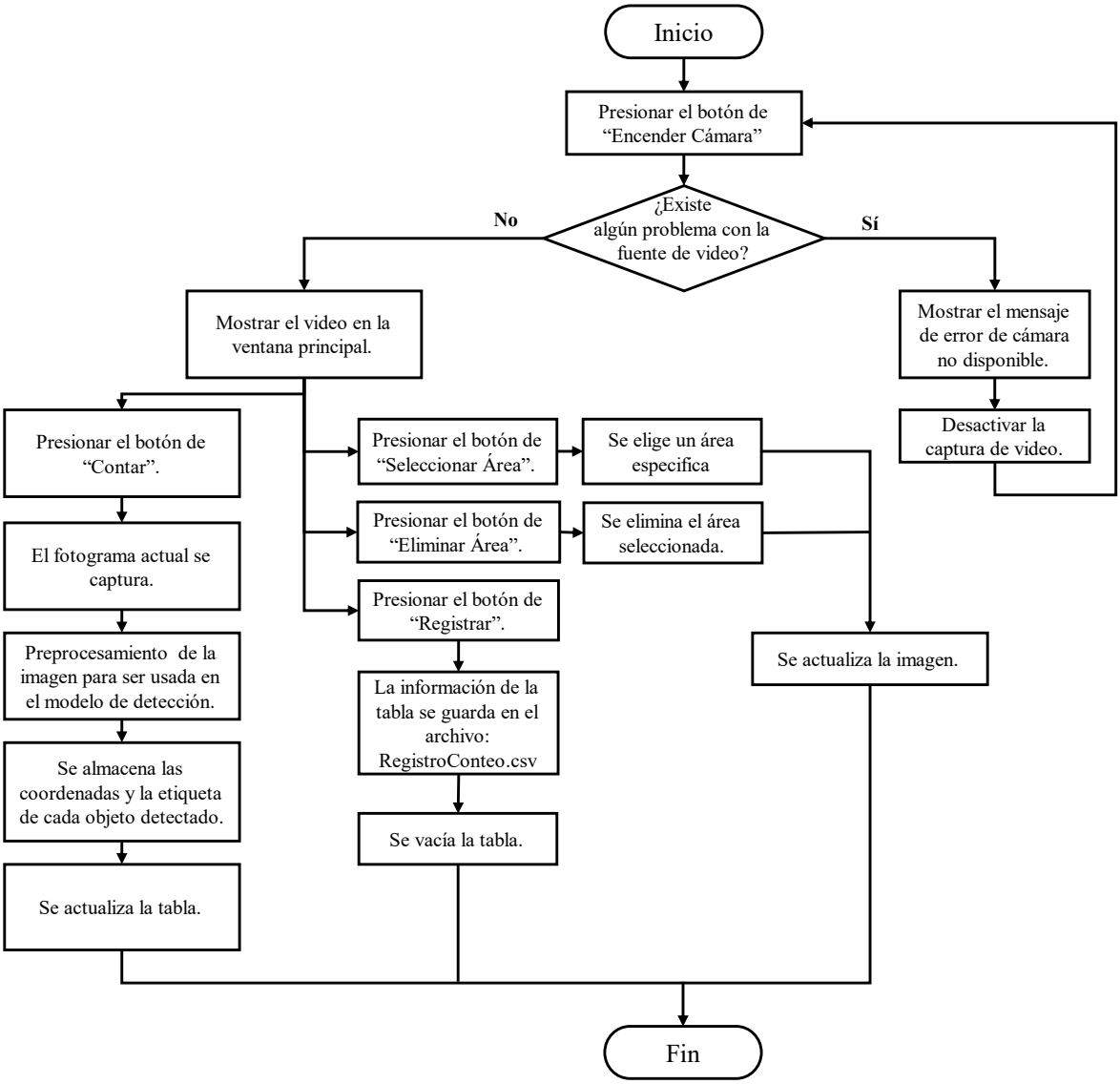

Figura 3.8: Diagrama de flujo de la ventana principal.

Capítulo IV

Implementación y pruebas

En esta sección se detalla el proceso seguido para desarrollar el sistema. Se implementan los diagramas de flujos del capítulo anterior mediante programación en el lenguaje Python.

4.1. Implementación del modelo

Esta etapa, concluye generando un modelo de detección de objetos personalizado, en el formato TFlite.

4.1.1. Dataset

Como se menciona en el capítulo II, el dataset constituye la fase inicial para entrenar un modelo de detección de objetos. Las consideraciones empleadas en la creación de estos imágenes y etiquetas influyen en el resultado final de la detección de objetos.

4.1.1.1. Selección del dispositivo de captura

Existe una gran variedad de dispositivos dedicados a la captura de imágenes, que ofrecen características distintas en cuanto a calidad de imagen, resolución en píxeles, nivel de brillo, ruido, precio, etc. Para el caso particular de este trabajo, el dispositivo debe tener un buen desempeño en la ubicación escogida y ser económicamente accesible. El dispositivo de adquisición de imágenes se coloca sobre el mostrador de cobro del local. La elección de esta ubicación se debe a que, de esta manera se obtiene una mejor vista de los objetos a detectar como se muestra en la Figura 4.1.

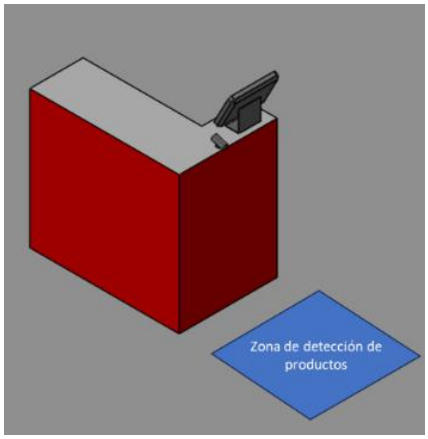

Figura 4.1: Ubicación del dispositivo de captura.

El dispositivo de adquisición de imágenes escogido es la cámara web modelo D BUGG FLY CAM que ofrece una resolución de 640x480 píxeles. Se comunica a la computadora mediante un cable de 1,5m con conector USB. La elección de este modelo se debe principalmente a las características físicas del mismo: peso reducido, movilidad, tamaño y a su relación de calidad-precio. El dispositivo se muestra Figura 4.2: Dispositivo escogido para la adquisición de imágenes [29].

Figura 4.2: Dispositivo escogido para la adquisición de imágenes [29].

Las principales características de esta cámara se detallan en la Tabla 4.1.

Tabla 4.1: Características del dispositivo de captura escogido [29].

Característica	**Descripción**
Conectividad	USB 2.0
Tipo de movimiento	Puede girar 360° y moverse cambiar de ángulo.
Tipo de enfoque	Tiene un enfoque manual
Peso	Alrededor de 50gr
Longitud del cable	1 metro.

4.1.1.2. Iluminación utilizada

Como se plantea en el capítulo II, se busca que la iluminación utilizada permita observar las características de los objetos a detectar, evitando la generación de sombras y reflejos.

En el entorno en donde se ubica este sistema, existen 2 fuentes de iluminación, la iluminación ambiental natural y la iluminación artificial led. Se observa que ambos tipos de iluminación ofrecen una buena apreciación de las características ya que entre las dos se complementan. Por lo tanto, se contempla estos dos tipos de iluminación en la creación del conjunto de datos. Además, se opta por actualizar la fuente de iluminación led por la mostrada en la Figura 4.3.

Figura 4.3: Fuente de iluminación led de 20W.

4.1.1.3. Especificación de los objetos a detectar

Una vez seleccionado el dispositivo de captura y establecidas las condiciones de iluminación y espacio, se procede a la adquisición de imágenes. Al revisar el trabajo disponible en[4], [5], [6], se observa que se utilizan 250 imágenes en el entrenamiento. Debido a que se busca un uso más general, se plantea utilizar un conjunto de 10 veces esa cifra, es decir, 2500 imágenes dividido en tres clases de objetos. Estas clases se se detallan a continuación:

- Botellas de gaseosas y agua: este tipo de conjunto de imágenes se usa para crear una categoría denominada “botella”, que consta de 1000 imágenes.
- Cajas de cartón medianas: Este tipo de productos se usa para la creación de la categoría denominada “caja”, que consta de 1000 imágenes.

- Caja de aceite Oro: Este tipo de caja es la más utilizada debido a que las fundas de aceite se venden con mayor frecuencia como se en la Tabla 4.2.Se identifica con la clase “caja1” y se usan 500 imágenes.

Tabla 4.2: Porcentaje de ventas en cajas de aceite vegetal.

ACEITE EN CAJAS	%VENTAS
ACEITE FUNDA PALMA DE ORO 400ML	34,3%
ACEITE LA FAVORITA 700ML	10,5%
ACEITE PALMA DE ORO 900ML	8,7%
ACEITE COCINERO 2L	7,2%
ACEITE FAVORITA DE 2 LT.	6,6%
ACEITE ACHIOTE FAVORITA 200ML	4,5%
ACEITE COCINERO 900ML	4,5%
ACEITE ACHIOTE PALMA DE ORO 200ML	4,5%
ACEITE CRIOLLO ACHOTE	4,0%
ACEITE ALESOL 937ML	3,9%
ACEITE ALESOL FUNDA	2,7%
ACEITE ACHIOTE FAVORITA 500ML	1,9%
ACEITE COCINERO 5L	1,7%

4.1.1.4. Etiquetado de las imágenes

En esta etapa se etiquetan todas las imágenes obtenidas en la etapa anterior, los archivos resultantes se organizan dentro una nueva carpeta y se comprime para facilitar su uso.

Como se menciona en el capítulo II, **Make Sense AI**, es la herramienta escogida para crear los archivos XML necesarios para el entrenamiento. La principal ventaja es la rapidez, ya que las imágenes son cargadas directamente desde el disco y no se requiere una conexión de banda ancha ya que no se cargan a la nube. Después de haber cargado las imágenes, se debe escoger la primera opción “Object Detection”, tal como se ve en la Figura 4.4.

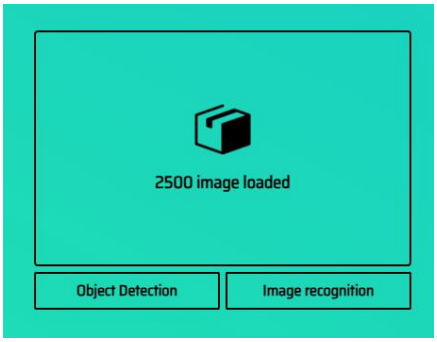

Figura 4.4: Opciones de Make Sense AI.

Después, se configura el nombre de las etiquetas para una mejor visualización. También es recomendable escoger colores llamativos para diferenciar cada clase tal como se muestra en la Figura 4.5.

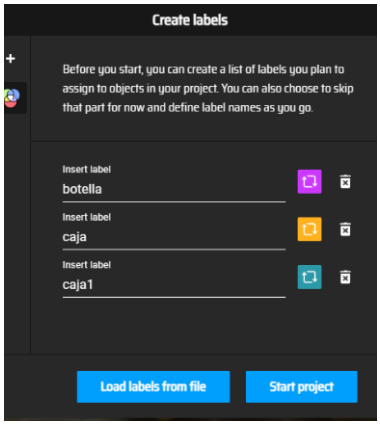

Figura 4.5: Creación de las etiquetas.

Luego se procede a etiquetar usando recuadros, es importante seleccionar la categoría correcta, esta interfaz es fácil de utilizar, permite la visualización de las imágenes al costado izquierdo y en el costado derecho se encuentra todas las etiquetas realizadas, al pasar el mouse por encima de ellas se muestra en qué posición está cada una, esto ayuda a orientarse cuando son muchas etiquetas, como se aprecia en la Figura 4.6.

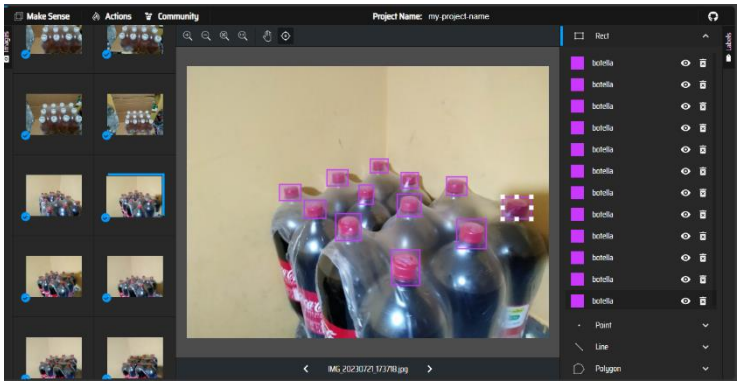

Figura 4.6: Etiquetado del conjunto de imágenes.

Al terminar de etiquetar todas las imágenes se debe exportar el archivo que contiene las etiquetas, la desventaja de esta aplicación es que el trabajo no se puede guardar automáticamente, por lo cual cuando se desee pausar el trabajo, se debe descargar las etiquetas, para retomar el punto en donde se quedó o modificar las posiciones de los recuadros se puede hacer uso de la opción importar anotaciones.

El formato en el que se guardan las etiquetas puede ser escogido entre los que se muestran en la Figura 4.7.

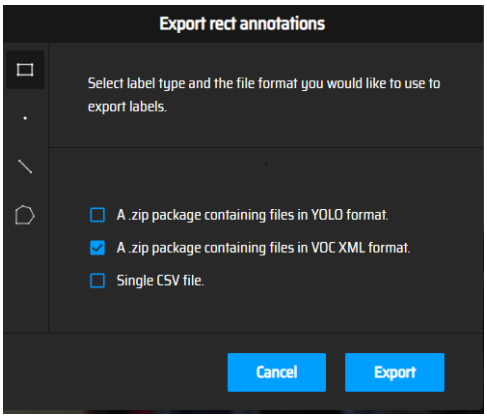

Figura 4.7: Opciones de formatos.

Ejemplo de formato **XML**:

```
<bndbox>
    <xmin>109</xmin>
    <ymin>128</ymin>
    <xmax>237</xmax>
    <ymax>265</ymax>
```

Ejemplo de formato **YOLO**:

```
1 0.057323 0.048708 0.042412 0.033797
```

Para finalizar, las imágenes y las etiquetas se colocan en la misma carpeta, que se comprime en un archivo zip, para ser cargado en Google Drive, de esta manera se ahorra tiempo, ya que cargar directamente al notebook de Google Colab se demora más tiempo.

4.1.2. Entrenamiento del modelo

Para entrenar el modelo en TensorFlow se utiliza la guía proporcionada en [30].

Se inicia creando un nuevo Notebook en Google Colab y se cambia el entorno de ejecución y se escoge la tarjeta GPU T4, es necesario crear las carpetas mediante el código mostrado en la Figura 4.8. Luego, se clona el repositorio desde GitHub. Este contiene los modelos y las librerías de TensorFlow necesarias, esto se consigue mediante el uso del comando mostrado en la Figura 4.9.

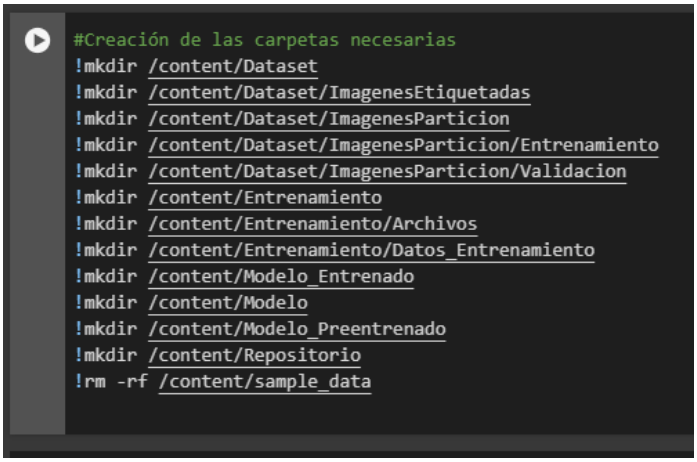

Figura 4.8: Creación del directorio.

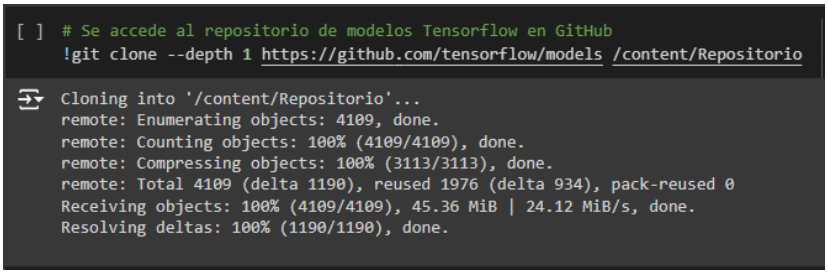

Figura 4.9: Clonación del repositorio de TensorFlow.

A continuación, se deben compilar los archivos usando el comando en Bash *protoc* (se usa para indicar que se está trabajando en un entorno de Python), luego es necesario instalar los requisitos de "setup.py", esto puede tardar un momento debido a que son algunos componentes (Figura 4.10).

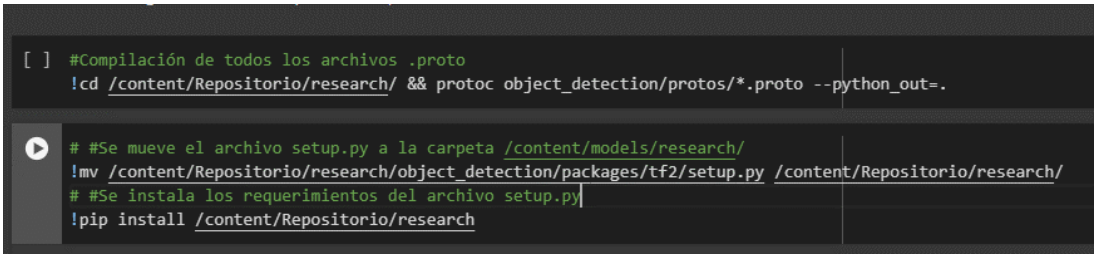

Figura 4.10: Instalación de requisitos de setup.py.

Para acceder a la carpeta de imágenes en Google Drive se usa la librería *drive*, que crea la carpeta "Drive" en donde se encuentran todos los archivos de la cuenta de Google, por ello es importante prestar atención para evitar borrar carpetas o archivos inintencionalmente, luego se

debe descomprimir el archivo en la carpeta llamada ImagenesEtiquetadas usando los comandos de la Figura 4.11.

```
#Se accede a Google Drive para cargar el archivo .zip de las imagenes y las etiquetas
from google.colab import drive
drive.mount('/content/Drive')

Mounted at /content/Drive

#Se copia el archivo ImagenesEtiquetadas a la carpeta /content
!cp /content/Drive/MyDrive/DATASET/ImagenesEtiquetadas.zip /content/Dataset

#Se descomprime el archivo zip en Dataset/ImagenesEtiquetadas
!unzip -q /content/Dataset/ImagenesEtiquetadas.zip -d /content/Dataset/ImagenesEtiquetadas
```

Figura 4.11: Acceso al conjunto de imágenes y etiquetas.

Después, se distribuyen las imágenes en las carpetas de Entrenamiento y Validación, en estas carpetas se colocan las imágenes al azar cumpliendo una relación en la que para el entrenamiento se coloca el 80% del total de las imágenes y para la validación se usa el 20%. Esto se consigue al usar el código disponible en los anexos en GitHub y como resultado se obtiene lo mostrado en la Figura 4.12.

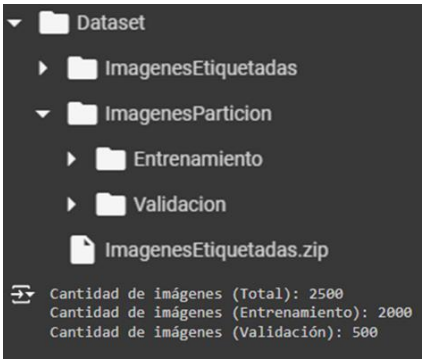

Figura 4.12: División del conjunto general de imágenes.

Luego, se debe crear el archivo en el que se encuentran todas las categorías a detectar, este archivo es de texto, pero ocupa un formato distinto, mediante el código de la Figura 4.13.

```
[ ]  #Creacion del archivo .pbtxt en donde se encuentran todos los objetos a detectar
     etiquetas = [
         {'id': 1, 'name': 'botella'},
         {'id': 2, 'name': 'caja'},
         {'id': 3, 'name': 'caja1'},
     ]
     with open('/content/Entrenamiento/Archivos/etiquetas.pbtxt', 'w') as file:
         for label in etiquetas:
             file.write(f"item {{\n    id: {label['id']}\n    name: '{label['name']}'\n}}\n")
```

Figura 4.13: Código utilizado para crear el archivo pbtxt.

Después, es necesario agrupar el contenido de los archivos XML de las carpetas de Entrenamiento y Validación en archivos CSV, que son utilizados para crear dos archivos con la extensión ".tfrecord". Estos archivos se utilizan en TensorFlow, al momento de entrenar el modelo, el formato de archivo es especifico ya que están diseñados para almacenar la información del conjunto de las imágenes etiquetadas, permitiendo un acceso eficiente a ellos [31].

El código de encuentra en el apéndice de Anexos, al ejecutar, se crean dos archivos en la carpeta Imágenes que corresponden a los CSV, como se muestra en la Figura 4.14.

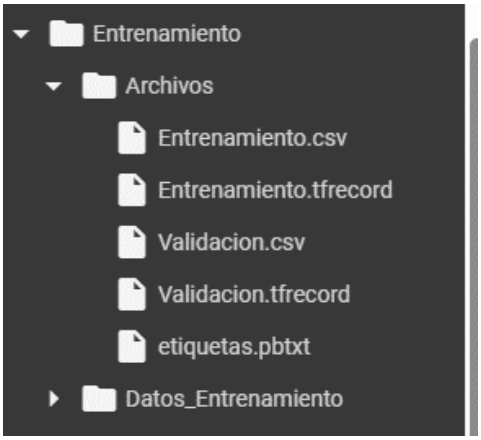

Figura 4.14: Creación de los archivos csv y tfrecords.

En este trabajo, se utiliza el modelo preentrenado: *ssd_mobilenet_v2_fpnlite_320x320*, principalmente debido a que ayuda a cumplir el primer requisito básico del sistema, las principales características se presentan en la revisión literaria.

Para ser usado en Google Colab, se lo descarga desde el repositorio de TensorFlow 2 Detection Model Zoo, para ello se ocupa el código mostrado en la Figura 4.15.

```
[ ]  # Descargar el modelo ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8
     %cd /content/Modelo_Preentrenado
     !wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
     !tar -xzf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz --strip-components=1 -C /content/Modelo_Preentrenado/
     !rm ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz

     # Descargar el archivo de configuración del modelo
     !wget https://raw.githubusercontent.com/tensorflow/models/master/research/object_detection/configs/tf2/ssd_mobilenet_v2_fpnlit
```

Figura 4.15: Descarga desde el repositorio del modelo preentrenado.

Después de esto, se configura el archivo “pipeline.config”, en este archivo constan las siguientes configuraciones: números de pasos que se va a entrenar el modelo, el número de clases a detectar, la ubicación de los archivos “tfrecord” y número de objetos a detectar. Se usan 65000 pasos para entrenar el modelo, los resultados se almacenan en la carpeta Datos_Entrenamiento. El proceso suele tardar aproximadamente unas 6 horas, al finalizar en la carpeta de entrenamiento se encontrarán los pesos denominados *checkpoints*, a partir de estos se puede generar el modelo tflite. Por defecto los modelos tienen un máximo de 10 objetos detectados por cada imagen que se ingresa, para cambiar esto se utiliza el código –max detections =100, en la Figura 4.16, se muestran los códigos utilizados para entrenar y generar el modelo.

```
#Entrenamiento
!python /content/Repositorio/research/object_detection/model_main_tf2.py \
--model_dir={"/content/Entrenamiento/Datos_Entrenamiento"} \
--pipeline_config_path= {"/content/Entrenamiento/Archivos/pipeline_customizado.config"} \
--num_train_steps={65000}
```

```
#Guardado y conversión a modelo .tflite
import zipfile
import tensorflow as tf

!python /content/Repositorio/research/object_detection/export_tflite_graph_tf2.py \
--max_detections=100 \
--pipeline_config_path {'/content/Entrenamiento/Archivos/pipeline_customizado.config'} \
--trained_checkpoint_dir {'/content/Entrenamiento/Datos_Entrenamiento'}  \
--output_directory {'/content/Modelo_Entrenado'}

modeloPB = tf.lite.TFLiteConverter.from_saved_model('/content/Modelo_Entrenado/saved_model')
modelotTfl = modeloPB.convert()

etiqueta = [etiqueta['name']+'\n' for etiqueta in etiquetas]

with zipfile.ZipFile('/content/ModeloV1.zip', 'w') as zipf:
    zipf.writestr('modelov1.tflite', modelotTfl)
    zipf.writestr('etiquetas.txt', etiqueta)
```

Figura 4.16: Entrenamiento y generación del modelo.

4.2. Implementación de la interfaz gráfica

La interfaz gráfica permite utilizar el modelo de detección de objetos, entrenado en la etapa anterior. El objetivo principal que cumple la interfaz es facilitar el proceso de captura de la imagen, detección y conteo de objetos, almacenamiento de los resultados en el registro. Se diseña considerando los requisitos identificados en el apartado 3.4. Para su desarrollo se utiliza el entorno de programación Python IDLE 3.9 y la librería Tkinter.

Para determinar las dimensiones de las ventanas, disposición y tamaño de los botones, se procede a obtener las características del dispositivo en el cual se prueba el sistema en condiciones reales de trabajo, las especificaciones se encuentran en la Tabla 4.3.

Tabla 4.3: Especificaciones del dispositivo.

Característica	**Descripción**
Pantalla	Pantalla LCD de 1024 x 763, relación de aspecto 4:3, dispone de entrada táctil.
Memoria RAM	2 tarjetas de 4GB de 2667 MHz.
Almacenamiento	disco sólido de 480 GB.
Procesador	Intel Celeron J6412 a 2GHz.
Sistema Operativo	Windows 10

4.2.1. Diseño y funciones de la interfaz gráfica

El diseño de las ventanas se basa en optimizar la visualización, se utiliza un fondo claro para contrastar con el color de los botones. El texto que se encuentra en cada botón está en negrita para resaltar. La fuente de texto utilizada en las ventanas es Times New Roman.

4.2.1.1. Ventana principal

La ventana principal tiene un tamaño de 1010x710 píxeles, cuenta con un icono en su esquina superior izquierda, se diseña para cumplir el requisito de estar optimizada al tamaño de la ventana. El video proveniente de la cámara se muestra en un espacio de 640x480 píxeles.

Los botones están posicionados en la parte superior para facilitar la interacción del usuario con la aplicación. El tamaño es adecuado, para permitir visualizar el texto que describe su función. El botón para acceder a las configuraciones tiene un icono y se encuentra situado en la esquina superior derecha. La separación entre los botones evita que sean presionados accidentalmente, estas características En la esquina inferior izquierda, existe un recuadro de texto, en el cual se brinda información, en el modo manual en este se busca el producto. En el costado derecho se encuentra una tabla en la que se registra la cantidad y productos detectados al presionar el botón de contar, la tabla cuenta con una barra de desplazamiento vertical que facilita la navegación. Estas características de muestran en la Figura 4.17.

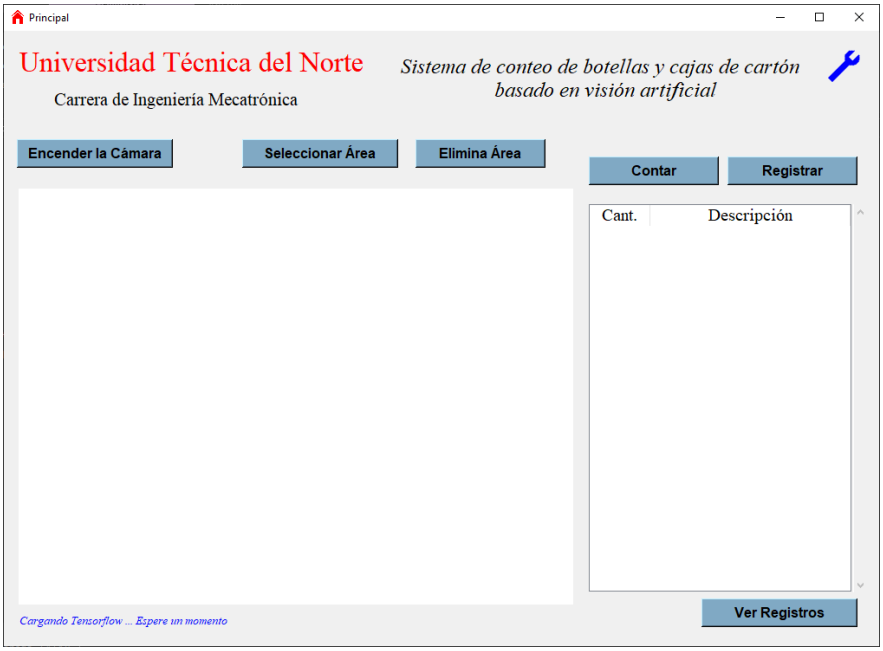

Figura 4.17: Ventana principal.

4.2.1.2. Venta de productos registrados

La ventana de Productos Registrados tiene un tamaño de 850x710 píxeles, cuenta con cuatro botones ubicados en la parte superior. Cada uno de estos, muestra el ingreso de productos en un periodo de tiempo predeterminado. El diagrama mostrado es de barras horizontales, esto sirve para comparar las cantidades de cada producto rápidamente. En la Figura 4.18. se muestran estas características.

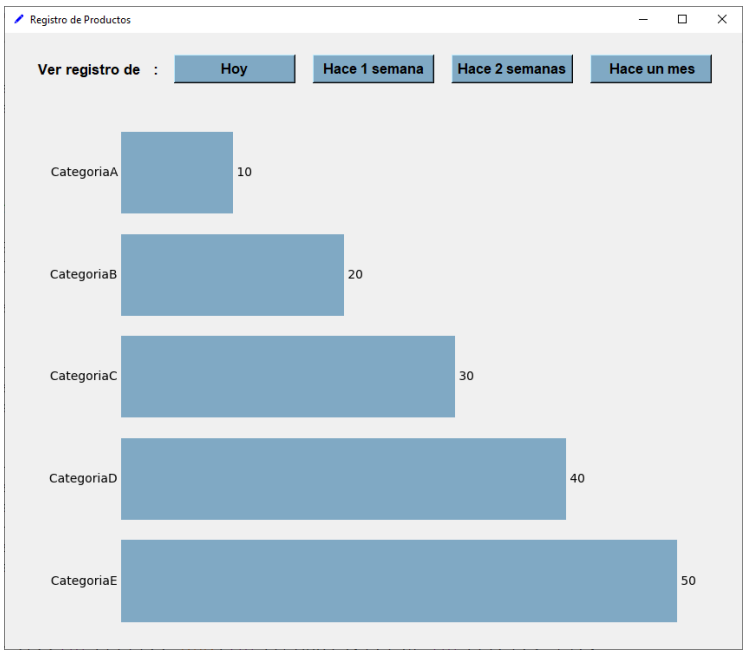

Figura 4.18: Ventana de visualización de los registros.

4.2.1.3. Ventana de configuración

La ventana de configuración del sistema cuenta con 5 campos que deben ser configurados por el usuario para que el sistema funcione correctamente. Para facilitar la ubicación del modelo y las etiquetas de las categorías en el directorio, se usan botones que al presionarlos abren el explorador de archivos de Windows y permiten una fácil selección de la ruta de los archivos.

Para configurar el umbral de detención y el tiempo en que se muestran los recuadros en la captura se utilizan dos controles deslizantes, cuyo valor se puede visualizar en el costado izquierdo conforme se mueven. Mediante una casilla, se activa el modo manual, en el cual se debe identificar el producto manualmente, esta opción se explica con detalle en el apartado 4.2.2. La disposición de estos campos se muestra en la Figura 4.19.

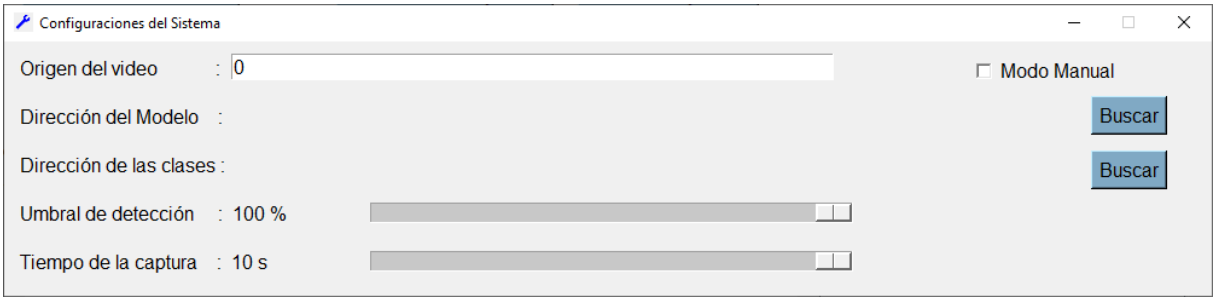

Figura 4.19: Ventana de configuración.

4.2.1.4. Funciones

La interfaz de usuario diseñada cuenta con las siguientes funciones que mejoran su utilización:

Origen del video: El origen del video mostrado en la pantalla puede ser de un dispositivo USB o de una cámara IP.

Direcciones de los archivos del modelo y clases: Al ser un campo configurable, permite que se pueda cambiar de modelo fácilmente.

Tiempo de captura: Se configura el tiempo en que se muestran los recuadros y el fotograma capturado, esto sirve para verificar si se han detectado todos los objetos.

Umbral de detección: Por defecto se usa el 85%, se cambia de acuerdo a la necesidad.

Ingreso de productos: Es una función en la cual se guardan los productos con una descripción y un identificador numérico.

Búsqueda de productos: Los productos se pueden encontrar de dos maneras, la primera se basa en utilizar el nombre, las coincidencias se muestran en la pantalla principal, la segunda es usar el indicador numérico, para esto es necesario contar con un lector de códigos de barras.

Eliminación de elementos mostrados en la tabla: Para borrar un elemento se debe dar clic derecho sobre este y escoger la opción de "Eliminar".

Corrección de color: Si se detecta un contraste bajo se realiza la ecualización de color en el preprocesamiento.

Ejecución de actividades en paralelo: Existen actividades que tardan un cierto tiempo en ser completadas. La carga de la librería de TensorFlow demora alrededor de 9s y la detección de objetos 1s. Se utiliza la librería *threading*, al iniciar el sistema, esto permite mostrar la ventana principal mientras se carga la librería de TensorFlow. También se utiliza al momento de usar el detector de objetos, permite que se muestre el vídeo en tiempo real mientras se esperan los resultados del modelo de detección.

4.2.2. Identificación de los productos

A excepción de la clase "caja1", las clases de "botella" y "caja" no se identifican directamente, ya que se entrenaron con diversos objetos de la misma clase. Para solucionar esto se añaden dos soluciones, la primera es la función de "Modo Manual". En esta función, el usuario puede ingresar el nombre del producto (botella o caja de cartón), para ser utilizado como etiqueta de los objetos detectados. Esto se puede ver en la Figura 4.20.

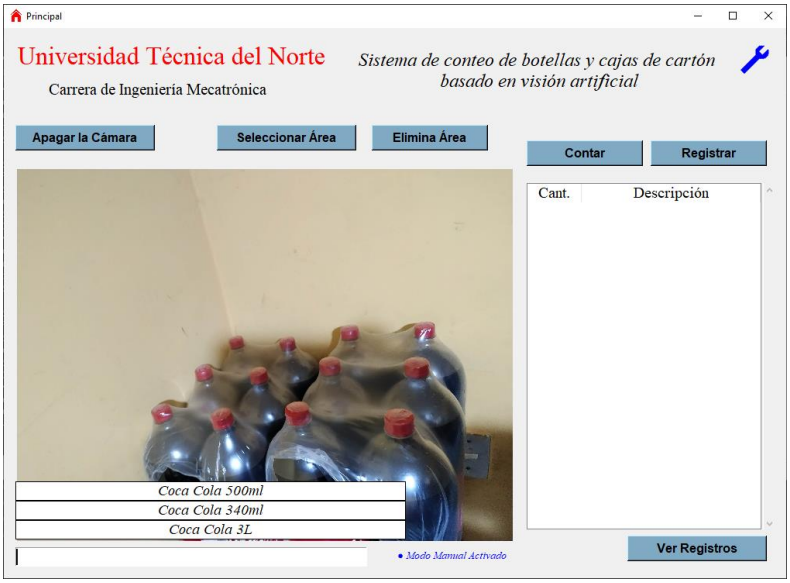

Figura 4.20: Interfaz de búsqueda de producto.

La segunda solución, es la identificación mediante el color de las tapas. Para determinar que productos se deben identificar automáticamente se basa de acuerdo el porcentaje de ventas que se muestra en la Tabla 4.4. Se observa que las ventas de agua representan, una parte considerable de las botellas, también esto se menciona en [32].

Tabla 4.4: Ventas de botellas de agua.

Botella de agua	%Ventas
DASANI SIN GAS 600ML	31,0%
TESALIA 625ML	50,6%
VILCAGUA DE 500ML	18,4%

En base a esto, se obtiene una muestra de 1000 colores de las tapas de cada producto mostrado en la Tabla 4.4, en la Figura 4.21, se muestran las primeras 200 muestras de cada tipo.

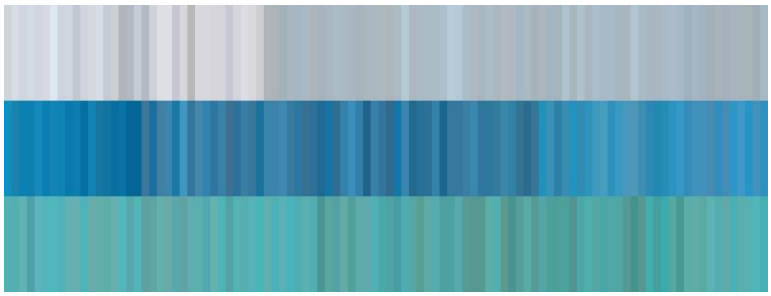

Figura 4.21: Muestras de color de tapas: Vilcagua, Dasani y Tesalia.

Después, estas muestras son ubicadas en los espacios de color RGB (Figura 4.22) y CIELAB (Figura 4.23). De esta forma se revisa de qué manera están distribuidos y se determina la mejor manera de compáralos y respectivamente.

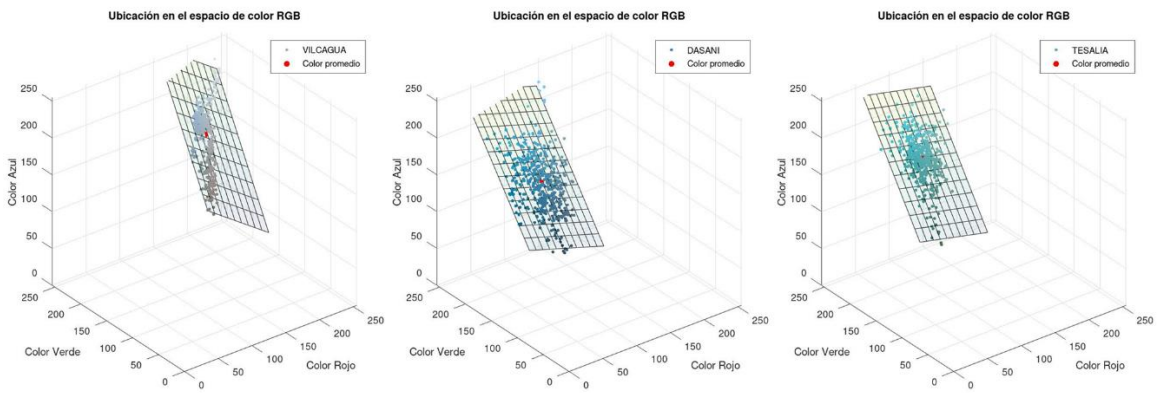

Figura 4.22: Ubicación de los colores en el espacio RGB

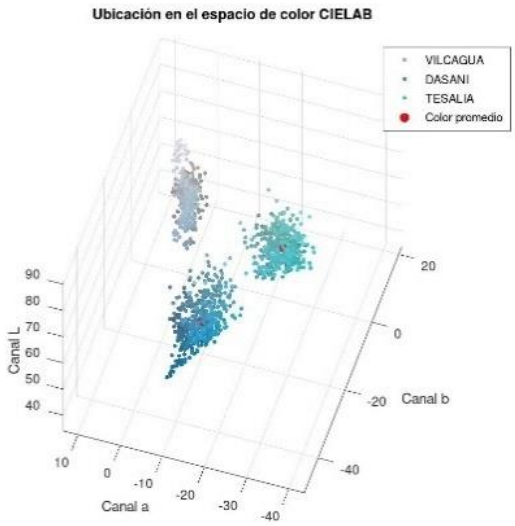

Figura 4.23: Ubicación de los colores en el espacio CIELAB.

En el espacio RGB, los colores de las tapas se ajustan mediante un plano, en cambio en el espacio CIELAB, los colores se agrupan en nubes de puntos. A partir de esto, se calcula en Octave los siguientes valores para cada grupo:

- Color promedio
- Coeficientes de los planos (mediante mínimos cuadrados)
- Distancia de cada con el plano ajustado (RGB).
- Distancia Delta E de cada color con respecto al color promedio (CIELAB).

Los resultados se ven en la Figura 4.24, en el caso de las distancias, se agrupan por valor mínimo, promedio y máximo.

```
Ventana de comandos
Nombre: VILCAGUA  Tamaño: 1000
ColorProm: [172,175,176]
Plano: -0.6804 * x + 1.6935 * y - z + -3.1655 = 0
Distancias Plano: [0.0003,2.4625,10.2732]
Distancias CIELAB: [1.3363,10.6513,28.9925]
Nombre: DASANI  Tamaño: 1000
ColorProm: [67,150,184]
Plano: -0.2401 * x + 1.1222 * y - z + 32.7332 = 0
Distancias Plano: [0.0003,3.0552,14.0223]
Distancias CIELAB: [0.7878,12.5806,28.6636]
Nombre: TESALIA  Tamaño: 1000
ColorProm: [96,178,176]
Plano: -0.2182 * x + 1.2377 * y - z + -23.1465 = 0
Distancias Plano: [0.0025,2.9735,12.5608]
Distancias CIELAB: [0.2478,11.6703,28.6803]
```

Figura 4.24: Resultados de los cálculos de color.

En la interfaz, se integra la función en la cual, se utiliza el modelo de detección para ubicar las tapas en la imagen. Luego, se realiza un recorte que cubre el 50% del recuadro. Con esto se calcula el color promedio y se compara con el color promedio de cada tipo de botella. La comparación de color se realiza en dos etapas: primero se calcula el valor Delta E en el espacio CIELAB, y luego, se calcula la distancia al plano en el espacio RGB. Esto se muestra en la Figura 4.25, en donde se almacenan los valores dentro del archivo de texto de las clases del modelo. En la Figura 4.26, se ve la función que busca el color.

```
botella
caja
Caja Aceite Palma de Oro $1
172,175,176,VILCAGUA,-0.6802,1.6935,-1,-3.1655,29,11
67,150,184,DASANI,-0.2401,1.1222,-1,32.7332,29,14
96,178,176,TESALIA,-0.2182,1.2377,-1,-23.1465,29,13
```

Figura 4.25: Archivo de texto de las clases.

```
def buscarColor(xmin,ymin,xmax,ymax):
    global ImgDetect,ValoresConfig
    centro=[int((xmax+xmin)*0.5),int((ymax+ymin)*0.5)]

    limite=[int((xmax-xmin)*0.25),int((xmax-xmin)*0.25)]

    imgCortada=ImgDetect[centro[1]-limite[1]:centro[1]+limite[1],
                         centro[0]-limite[0]:centro[0]+limite[0]]
    imgCortadaORG=ImgDetect[ymin:ymax,xmin:xmax]
    promedio_color = cv2.mean(imgCortada)
    bs_clrs=[]
    with open(ValoresConfig[1], 'r') as archivo:
        bs_clrstxt = archivo.readlines()
    bs_clrstxt=bs_clrstxt[1:]
    for clrs in bs_clrstxt:
        clrs=clrs.split(",")
        bs_clrs.append([int(clrs[0]),int(clrs[1]),int(clrs[2]),clrs[3],
                        float(clrs[4]),float(clrs[5]),
                        float(clrs[6]),float(clrs[7]),
                        int(clrs[8]),int(clrs[9]),int(clrs[10]),int(clrs[11]),
                        int(clrs[12]),int(clrs[13]),int(clrs[14])])
    return planoDD(bs_clrs,promedio_color[:3])
```

Figura 4.26: Implementación de búsqueda de id en base al color.

4.3. Implementación en condiciones reales de trabajo

Se copian los archivos desde GitHub o desde una memoria USB, y se instala Python 3.9 en el computador de la tienda, luego se ejecuta el archivo .bat que instala las librerías necesarias, también se crea un acceso directo para arrancar al sistema desde el escritorio. Al utilizarlo por primera vez se muestra el mensaje que pide actualizar las configuraciones, esto se evidencia en la Figura 4.27

En la Figura 4.28, se muestra la colocación y posicionamiento del dispositivo de captura en el mostrador, mediante la pinza de este y cinta de doble cara.

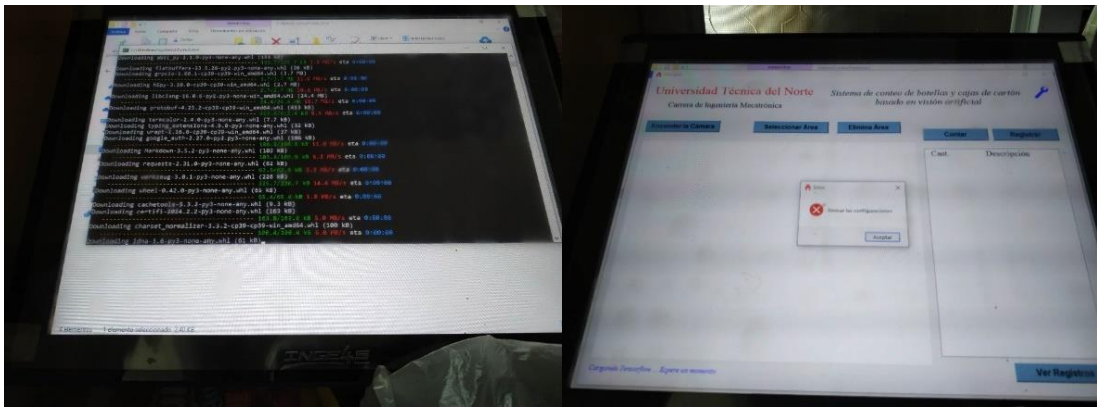

Figura 4.27. Instalación del sistema y ejecución por primera vez del programa.

Figura 4.28. Ubicación de la cámara.

4.3.1. Resultados de la detección

En la Figura 4.29, se muestra el funcionamiento del sistema detectando los objetos.

Figura 4.29: Detección de botellas y cajas de cartón.

4.4. Realización de pruebas

4.4.1. Resultados del entrenamiento

4.4.1.1. Estadísticas del entrenamiento

Mediante TensorBoard se obtienen las gráficas de las estadísticas del entrenamiento que se muestran en la Figura 4.30.

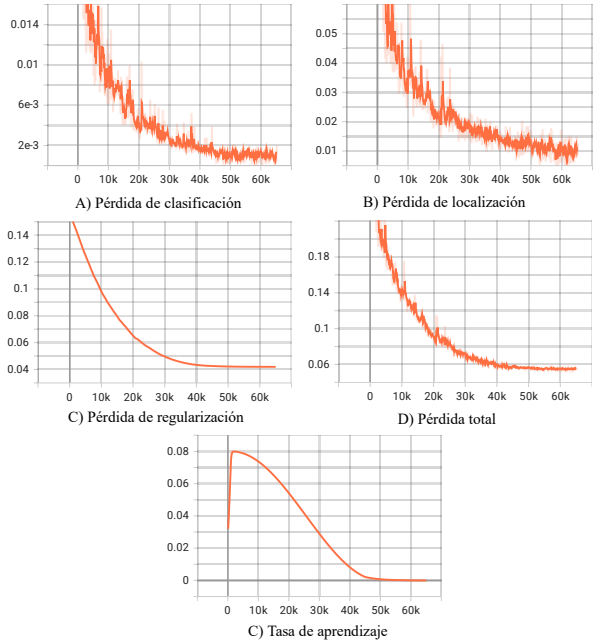

Figura 4.30: Estadísticas del entrenamiento del modelo.

Pérdida de clasificación: Es un valor que representa el error del modelo al clasificar los objetos detectados.

Pérdida de localización: Representa el error del modelo al localizar los objetos en la imagen.

Pérdida de regularización: Expresa el comportamiento del modelo con nuevos datos de entrada, evita que el modelo se solo funcione con la base de datos con la que fue entrenado.

Pérdida total: Es la suma de las demás perdidas se observa cómo empieza con un porcentaje alto de error y a medida que se entrena más el modelo, va disminuye hasta converger a un valor aproximado de 0,06.

Tasa de aprendizaje: Es un parámetro que controla cuánto se ajusta el modelo a las predicciones con cada nueva imagen. Al inicio la tasa es muy alta, el modelo puede aprender rápido, pero de manera inexacta. Al final, tiende a acercarse a 0, es decir, el aprendizaje es más lento, pero con más precisión.

Por lo tanto, se puede inferir que el entrenamiento del modelo se realizó de la manera correcta y la cantidad de pasos y de imágenes fue la suficiente.

4.4.1.2. Matriz de confusión

Es base a una muestra de 200 imágenes se obtuvo los siguientes resultados presentados en la Figura 4.31.

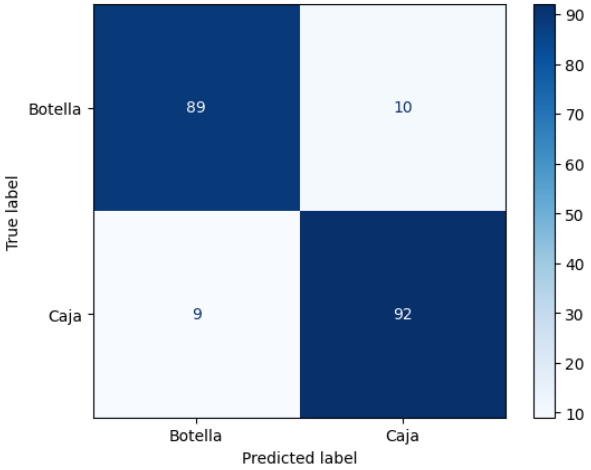

Figura 4.31: Matriz de confusión.

Se calculan los siguientes parámetros:

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{89 + 92}{89 + 92 + 9 + 10} = 0{,}905 \rightarrow 90{,}5\%$$

$$\text{Precisión} = \frac{TP}{TP + FP} = \frac{89}{89 + 9} = 0{,}908 \rightarrow 90{,}8\%$$

$$\text{Recall(Sensibilidad)} = \frac{TP}{TP + FN} = \frac{89}{89 + 10} = 0{,}898 \rightarrow 89{,}8\%$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}} = 2 \cdot \frac{0{,}905 \cdot 0{,}898}{0{,}905 + 0{,}898} = 0{,}901 \rightarrow 90{,}1\%$$

Al interpretar los valores se puede afirmar que el modelo tiene valores altos en cuanto a exactitud, precisión, sensibilidad y equilibrio. Esto significa que el sistema detecta adecuadamente a cada clase de objeto, evitando crear falsos positivos, en conclusión, el parámetro más significativo en el contexto de este trabajo es la sensibilidad, ya que, si se detectan correctamente todos los objetos de interés en la imagen, el conteo es preciso y válido.

Capítulo V

Conclusiones, recomendaciones y trabajo a futuro

5.1. Conclusiones

- Las condiciones óptimas para la adquisición de imágenes de botellas y cajas contemplan las características del dispositivo de captura seleccionado, el tipo de fuente de iluminación utilizada y la configuración del escenario de captura. En el caso particular de estos dos objetos, se tiene que, la iluminación no debe generar sombras ni saturar la imagen, la posición de los objetos debe permitir la visualización de la mayor parte de sus características, siendo la forma y el color las principales.

- El diseño del algoritmo del sistema se centra principalmente, en integrar y desarrollar un modelo de detección de objetos y una interfaz de usuario. Las condiciones de espacio e iluminación se toman en cuenta en la creación de funcionalidades que permiten una respuesta rápida y fluida.

- La implementación del sistema se llevó a cabo rápidamente, debido a las consideraciones tomadas en la etapa de diseño, mediante la realización de pruebas se pudo validar la eficacia en condiciones de trabajo reales, obteniendo una precisión de 90,8% en el conteo de botellas y cajas de cartón.

5.2. Recomendaciones

- Se recomienda que el dispositivo de captura tenga una buena respuesta en diversos escenarios y condiciones de iluminación, de manera que se pueda obtener las características de los objetos a detectar con un buen nivel de detalle.

- Al utilizar software de código abierto, es importante tener en cuenta la licencia y las posibles limitaciones de garantía. En determinados casos, pueden surgir problemas debido a incompatibilidades o falta de soporte.

- Es recomendable guardar los *checkpoints*, del entrenamiento ya que esto permite retomar el punto guardado en cualquier otro momento.

5.3. Trabajo a futuro

- Profundizar la investigación sobre la implementación de técnicas de visión artificial en la automatización de otros procesos dentro del concepto de tiendas inteligentes.

- Utilizar el seguimiento de objetos o *tracking* para detectar el ingreso y salida de mercadería de manera automática.

- Implementar sistemas similares al presente en dispositivos móviles como teléfonos inteligentes, que integren bases de datos en la nube.

Bibliografía

[1] Ministerio de Producción Comercio Exterior Inversiones y Pesca, “Tiendas de barrio son aliados importantes en medio de emergencia sanitaria”, febrero de 2020.

[2] Grupo Bit, “La importancia de las tiendas de barrio”, febrero de 2020.

[3] E. Tapia, “Un 35% de contribuyentes ya usa facturas electrónicas”, noviembre de 2022.

[4] A. Tonioni, E. Serra, y L. Di Stefano, “A deep learning pipeline for product recognition on store shelves”, en *2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, IEEE, dic. 2018, pp. 25–31. doi: 10.1109/IPAS.2018.8708890.

[5] J. Bailón y M. Rodríguez, “Desarrollo de un sistema reconocedor de frutas para supermercados aplicando visión artificial”, Universidad de Guayaquil Facultad de Ciencias Matemáticas y Físicas Carrera, 2019.

[6] A. Fernández, “SIRP: Sistema Inteligente de Reconocimiento de Productos”, 2019.

[7] J. Moreno y J. Vera, “Diseño y simulación de estanterías inteligentes mediante el uso de inteligencia artificial”, ESPOL, 2020.

[8] E. Morán, “Diseño e implementación de una red neuronal convolucional para reconocimiento de productos de una empresa de retail”, Espol, 2018.

[9] D. Mero y M. Velásquez, “Gestión de inventarios y su incidencia en las compras, caso: Emprendimiento de víveres ubicados en la parroquia Manta”, *593 Digital Publisher CEIT*, vol. 8, núm. 1, pp. 174–187, ene. 2023, doi: 10.33386/593dp.2023.1.1552.

[10] J. Mira y D. Soler, *Manual del transporte de mercancías*. en Biblioteca de logística. Marge Books, 2024. [En línea]. Disponible en: https://books.google.com.ec/books?id=AWJUDQAAQBAJ

[11] Wikipedia contributors, “Plastic bottle — Wikipedia, The Free Encyclopedia”, 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Plastic_bottle&oldid=1227897011

[12] X. L. Gallo, S. Lechon, S. Mora, y D. Vallejo-Huanga, “MARRSIDS: Monitoring Assistant to Reduce the Risk of Sudden Infant Death Syndrome”, en *2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA)*, IEEE, abr. 2019, pp. 1–4. doi: 10.1109/STSIVA.2019.8730261.

[13] J. O’Connor, “Undercover Algorithm: A Secret Chapter in the Early History of Artificial Intelligence and Satellite Imagery”, *International Journal of Intelligence and CounterIntelligence*, pp. 1–15, jun. 2022, doi: 10.1080/08850607.2022.2073542.

[14] W. E. Snyder y H. Qi, *Fundamentals of Computer Vision*. Cambridge University Press, 2017. doi: 10.1017/9781316882641.

[15] A. Pujare, P. Sawant, H. Sharma, y K. Pichhode, “Hardware Implementation of Sobel Edge Detection Algorithm”, *ITM Web of Conferences*, vol. 32, p. 03051, jul. 2020, doi: 10.1051/itmconf/20203203051.

[16] V. Wiley y T. Lucas, “Computer Vision and Image Processing: A Paper Review”, *International Journal of Artificial Intelligence Research*, vol. 2, núm. 1, p. 22, jun. 2018, doi: 10.29099/ijair.v2i1.42.

[17] İ. ÇINAR, M. KOKLU, y Ş. TAŞDEMİR, “Kuru Üzüm Tanelerinin Makine Görüşü ve Yapay Zeka Yöntemleri Kullanılarak Sınıflandırılması”, *Gazi Journal of Engineering Sciences*, vol. 6, núm. 3, pp. 200–209, dic. 2020, doi: 10.30855/gmbd.2020.03.03.

[18] D. Atchison, *Optics of the Human Eye*. New York: CRC Press, 2023. doi: 10.1201/9781003128601.

[19] Y.-Y. Lai, “What do CCT, CIE, and SPD mean in LED lighting?”, julio de 2021. [En línea]. Disponible en: https://luminusdevices.zendesk.com/hc/en-us/articles/4403685063437-What-do-CCT-CIE-and-SPD-mean-in-LED-lighting

[20] S. Karki *et al.*, “Classification of strawberry ripeness stages using machine learning algorithms and colour spaces”, *Hortic Environ Biotechnol*, vol. 65, núm. 2, pp. 337–354, abr. 2024, doi: 10.1007/s13580-023-00559-2.

[21] V. Castillo, “‘Diseño evolutivo de arquitecturas de Deep Learning para detección de vías de transporte’ - Tesis Fín de Máster”, 2019.

[22] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, y H. Arshad, “State-of-the-art in artificial neural network applications: A survey”, *Heliyon*, vol. 4, núm. 11, p. e00938, nov. 2018, doi: 10.1016/j.heliyon.2018.e00938.

[23] V. Wiley y T. Lucas, “Computer Vision and Image Processing: A Paper Review”, *International Journal of Artificial Intelligence Research*, vol. 2, núm. 1, p. 22, jun. 2018, doi: 10.29099/ijair.v2i1.42.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, y L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, ene. 2018.

[25] Y. Chandola, J. Virmani, H. S. Bhadauria, y P. Kumar, “Lightweight end-to-end Pre-trained CNN-based computer-aided classification system design for chest radiographs”, en *Deep Learning for Chest Radiographs*, Elsevier, 2021, pp. 167–183. doi: 10.1016/B978-0-323-90184-0.00001-1.

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, y S. Belongie, “Feature Pyramid Networks for Object Detection”, dic. 2016.

[27] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector”, dic. 2015, doi: 10.1007/978-3-319-46448-0_2.

[28] P. Pasuy, “Clasificación de aguacates basado en visión por Computador”, 2019.

[29] BAAM, “Camara web dbugg PC11 480p”, junio de 2021, *BAAM*.

[30] L. Vladimirov, “TensorFlow 2 Object Detection API tutorial”, 2020.

[31] V. B. Durdi, A. Kiran, A. Rao, S. S. Bhat, B. Santhosh, y M. Kumar, “A Novel Method for Recognizing Hand Gestured Sign Language Using the Stochastic Gradient Descent Algorithm and Convolutional Neural Network Techniques”, *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, núm. 7s, pp. 529–538, 2024.

[32] M. N. Castro Cárdenas, K. A. Cedeño Hidalgo, J. O. Zurita Cueva, y others, “Diseño de un Plan de Negocios para la comercialización de agua purificada en botellas de papel en la ciudad de Guayaquil y sus alrededores”, ESPOL. FCSH., 2018.

Anexos

Códigos y scripts disponibles en: https://github.com/CrisCC27/Mecatronica-2024