Sistema de reconocimiento y conteo de productos de panadería

Brayan David Ramos Caicedo

Informe de semestre de industria para optar al título de Ingeniero Electrónica otorgado por la Universidad de Antioquia

Asesor
David Stephen Fernández Mc Cann, Ph.D

Universidad de Antioquia
Facultad de ingeniería
Ingeniería Electrónica
Medellín, Antioquia, Colombia
2022

Cita	(Ramos Caicedo, 2022)
Referencia	Ramos Caicedo, B. D. (2022). *Sistema de reconocimiento y conteo de productos de panadería* [Semestre de industria]. Universidad de Antioquia, Medellín.
Estilo APA 7 (2020)	

Contenido

Resumen

Con el fin de optimizar procesos y brindar a los usuarios experiencias agradables, algunas empresas se dedican a generar soluciones tecnológicas para facilitar toda la gestión relacionada al negocio de dichos usuarios. CincoE S.A.S provee especialmente módulos de tecnologías para administración gerencial a las empresas que contratan sus servicios.

El proyecto desarrollado durante el periodo de semestre de industria y enunciado en el presente informe explora el procesamiento digital de imágenes y el uso de Deep Learning para facilitar el trabajo de los empleados en la empresa de De Lolita, permitiéndoles la clasificación y conteo de productos de panadería al finalizar la jornada laboral con fines contables internos.

Para lograr lo propuesto se escogieron 5 productos de panadería: aromática, alfajor de maicena, alfajor de chocolate, parmesana y corazón. Para cada producto se tomaron 20 imágenes y se llevó a cabo un proceso de data augmentation, obteniendo así un dataset de 100 imágenes por producto que se usaron para entrenar con el algoritmo YOLOv5.

Como resultado se tiene una aplicación móvil desarrollada en React Native que permite tomar foto o subir una ya almacenada en el dispositivo móvil, enviarla a un backend desarrollado en Python con el modelo entregado por YOLOv5 y, finalmente, clasificar los 5 tipos de productos usados por la empresa De Lolita y mostrar el un listado de cada producto detectado en la imagen, así como la cantidad de cada uno.

1. Introducción

Destinar tiempo al final de la jornada laboral para realizar tareas que requieran un esfuerzo físico o mental puede llevar a errores y agotamiento por parte del empleado. En particular, el conteo de productos alimenticios para fines contables en un restaurante o puesto de comida al final de una larga jornada conlleva a una alta probabilidad de error por el cansancio asociado a la carga laboral. La realización de un sistema de reconocimiento y conteo de dichos productos tiene un gran impacto económico y en el bienestar del trabajador, puesto que es más rentable que el procedimiento manual, permite ahorrar tiempo y evita un esfuerzo adicional al empleado.

La estimación de objetos para su conteo a partir de imágenes es una tarea retadora debido a una serie de razones, entre ellas la variabilidad en la apariencia debido a la iluminación o la oclusión, debido a los demás objetos que los rodean. Algoritmos previos de conteo de objetos, frutas, por ejemplo; con base en métodos de visión por computadora han dado buenos resultados, pero gran parte de estos son afectados por los problemas mencionados anteriormente y requieren de un ambiente cuidadosamente controlado para su correcto funcionamiento.

Deep learning es una buena elección para tratar con los problemas que puedan presentarse al usar técnicas de procesamiento digital de imágenes o visión por computadora, teniendo en cuenta los grandes logros que ha alcanzado en cuanto a clasificación y detección de objetos en imágenes. Adicional a esto, permite ahorrar tiempo en cuanto al tratamiento o procesamiento de la base de datos, puesto que no requiere una previa extracción de características ya que la propia red se encarga de esto.

La propuesta planteada consiste en aplicar técnicas de procesamiento digital de imágenes y algoritmos de *Deep Learning* para reconocer los productos alimenticios ofrecidos por el cliente y poder hacer un conteo de estos de forma ágil y rápida. Esto permitirá ahorrar tiempo destinado a dicha labor y evitará un esfuerzo adicional al empleado.

2. Objetivos

2.1. Objetivo general:

Implementar un sistema para el reconocimiento y conteo de diferentes tipos de productos de panadería mediante el uso de técnicas de procesamiento digital de imágenes y *Deep Learning*, a fin de disminuir la carga laboral y facilitar la realización de dicha labor.

2.2. Objetivos específicos:

1. Construir una base de datos con suficientes muestras para entrenar el modelo de *Deep Learning*. Dicha base de datos se construirá tomando fotos de 5 tipos de productos, por lo que se tendrán 5 clases entre productos de panadería y bebidas.

2. Realizar el preprocesado de las imágenes contenidas en la base de datos.

3. Hacer uso de un modelo previamente entrenado de *Deep Learning* como punto de partida para el entrenamiento del sistema (transferencia de aprendizaje), y evaluar su desempeño.

4. Realizar pruebas del sistema desplegado como una aplicación móvil o página web que permita tomar fotos a las que se les aplicará el modelo obtenido previamente para realizar el conteo en un escenario real, con el fin de evaluar su desempeño en dicho ambiente.

3. Marco teórico

3.1. Inteligencia artificial

El aprendizaje automático, también conocido como machine learning, es un campo de la inteligencia artificial que permite a las máquinas predecir futuras entradas en base a una información previa recolectada usada para su entrenamiento. El Machine learning ha tenido una gran relevancia al momento de hacer análisis de datos, puesto que a través de unas muestras de entrenamiento que brindan características relevantes sobre la clasificación del fenómeno, se descubren patrones para lograr predecir futuras muestras que ingresen al sistema.

Un sistema de aprendizaje de un algoritmo de machine learning se puede dividir en tres partes principales: Un proceso de decisión en el cual el algoritmo genera una estimación sobre un patrón de datos basándose en datos de entrada, una función de error que permite evaluar la predicción del modelo y un proceso de optimización de modelo que permite ajustar parámetros de este para que se adapte mejor, disminuyendo la discrepancia entre el ejemplo conocido y la estimación del modelo.

La estructura de un sistema que hace uso de un algoritmo de machine learning consiste en el uso de 3 tipos de capas que permiten llevar a cabo las tareas principales de dicho algoritmo. La primera de ellas es la capa de entrada, la cual recibe los datos y los pasa a una capa oculta que se encarga de analizar la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa, la cual puede ser otra capa oculta para realizar más procesamiento sobre los datos, algo demasiado útil para problemas no lineales. Adicionalmente, el resultado de una capa oculta puede pasar a una capa de salida que proporciona el resultado final de todo el procesamiento de datos que realiza el sistema. Una representación gráfica de este sistema se presenta en la **Figura 1**.

Al aumentar la escala y dificultad del análisis de datos, los métodos tradicionales de Machine learning empiezan a tener dificultades para lograr buenos resultados. Por esta razón, aparecen los modelos de Deep learning con una arquitectura más compleja que permite mejorar la capacidad en cuanto a análisis de datos, a través de la extracción automática de niveles de abstracción capa por capa. De esta forma, las características calculadas en una capa oculta dada pueden ser usadas en una capa oculta más profunda, lo que permite que la red neuronal explote la estructura compositiva de una función y se aproxime a muchas funciones con pocos pesos.

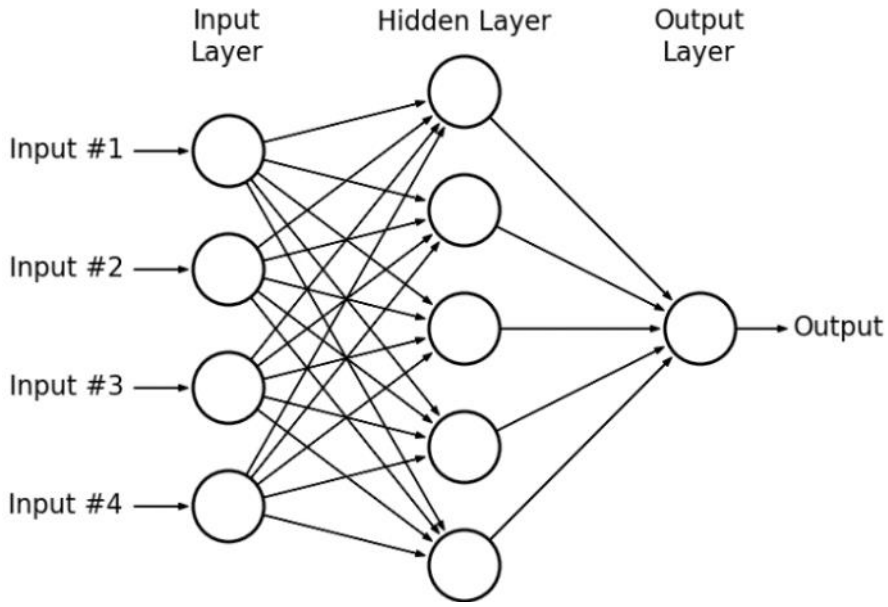

Figura 1. Ejemplo de una red neuronal con una capa oculta.

Dado que el proyecto consiste en el análisis de imágenes de productos de panadería, se decide usar Deep learning para el proceso de extracción de características, así como para el proceso de clasificación y su posterior conteo; esto por la eficiencia de los algoritmos de Deep learning para el tratamiento de imágenes ya que puede determinar el conjunto de características que distinguen diferentes categorías de datos entre sí.

La red de Deep learning usada por el modelo elegido es conocida como red neuronal convolucional o CNN por sus siglas en inglés, y toma este nombre por una operación matemática lineal entre matrices llamada convolución. Esta red tiene un excelente desempeño especialmente en problemas relacionados con imágenes, como lo son la clasificación, visión por computadora y procesamiento de lenguaje natural.

En particular, para obtener el modelo se hace uso del popular algoritmo You Only Look Once versión 5, conocido comúnmente como YOLOv5; un algoritmo de detección de objetos en imágenes que divide la imagen de interés en un sistema de cuadrículas en donde cada celda es responsable de detectar los objetos existentes en su interio. Este algoritmo es uno de los más populares puesto que permite extraer y determinar objetos al mismo tiempo, algo que sus predecesores hacían en dos o más pasos, obteniendo así un gran desempeño, en especial en tareas que requieren un tiempo de respuesta corto. La arquitectura de YOLOv5 consiste en una CNN con las

operaciones típicas de estas redes y un tensor de salida que entrega las posiciones de un cuadro delimitador que encierra cada objeto detectado, probabilidad de que haya un objeto y una probabilidad por cada clase definida. Una representación de esta arquitectura se muestra en la **Figura 2**.

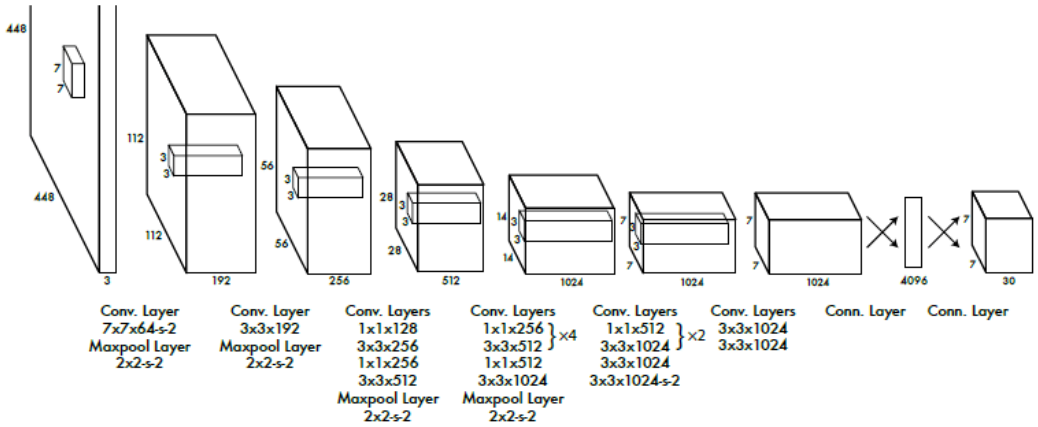

Figura 2. Arquitectura CNN de YOLO.

3.2. Data augmentation

El desempeño de la mayor parte de modelos de machine learning, y en especial de Deep learning, dependen de la calidad, cantidad y relevancia de los datos de entrenamiento [13]. En especial, la cantidad de datos es el problema más engorroso en estos sistemas dado que requiere de una gran cantidad de tiempo para su obtención.

Dada la base de datos relativamente pequeña para el proyecto, se opta por usar la técnica conocida como data augmentation, la cual consiste en un conjunto de técnicas para artificialmente incrementar la cantidad de datos de entrada haciendo uso de los datos ya existentes. Los datos nuevos se obtienen a partir de transformaciones sobre los datos iniciales, tales como rotación, aplicación de filtros, suavizado de imagen, escalamiento, transformaciones no lineales, etc. En la **Figura 3** se observa un ejemplo del uso de data augmentation sobre una imagen de entrada correspondiente a una mariposa.

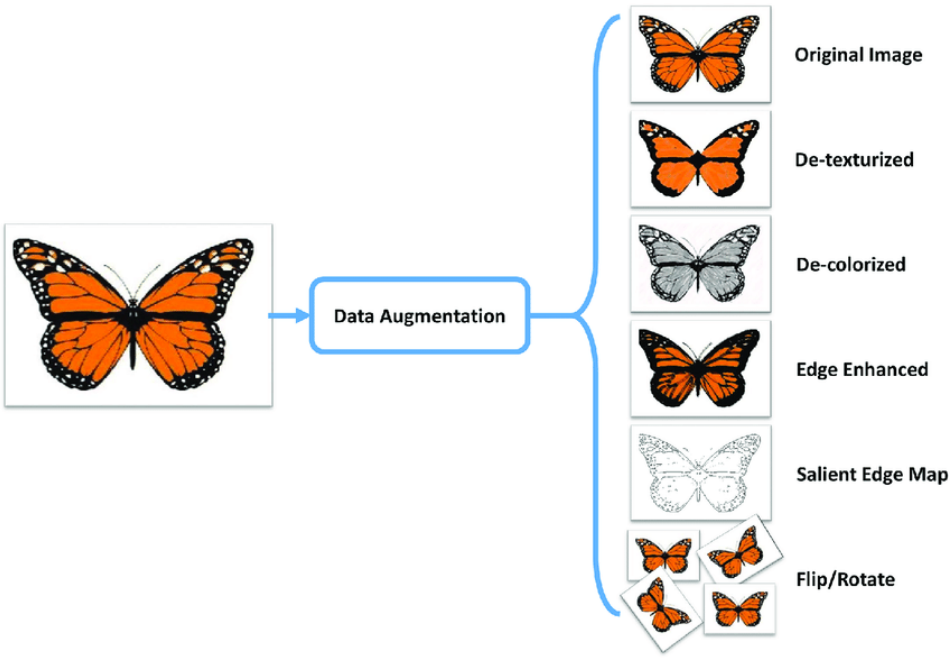

Figura 3. Ejemplo de data augmentation.

3.3. Python

Python es un lenguaje de programación interpretado, orientado a objeto y de alto nivel con semántica dinámica que debido a su estructura es muy atractivo para el desarrollo de aplicaciones de todo tipo de forma rápida, dada la facilidad de aprendizaje que tiene. Este lenguaje permite incluir una gran cantidad de librerías que facilitan la programación, entre las cuales se encuentran algunas usadas en el proyecto:

3.3.1. Numpy: Es un paquete fundamental para computación científica en Python e incluye una gran variedad de operaciones matemáticas para arreglos y otros tipos de datos.

3.3.2. OpenCV: Open Source Computer Vision Library u OpenCV es una librería de código abierto para visión por computadora y machine learning y contiene una enorme cantidad de algoritmos optimizados para ambas áreas.

3.3.3. FastAPI: FastAPI es un framework web moderno, rápido y de alto rendimiento para construir APIs con Python y está basado en las anotaciones de tipo estándar de Python. Su sintaxis fácil e intuitiva hace que sea un framework con una curva de aprendizaje muy rápida, permitiendo implementar una API en poco tiempo, sin perder robustez en el código o el rendimiento de la aplicación.

3.4. LabelImg

LabelImg es una herramienta gráfica desarrollada con Python y QT para realizar anotaciones en una imagen, útil para generar el dataset para entrenar el modelo de YOLOv5, puesto este requiere la imagen y un archivo de texto que contenga información de esta, tales como las coordenadas del cuadro delimitador que encierra al objeto de interés y la clase a la que pertenece.

3.5. React Native

React native es un framework que permite crear aplicaciones móviles con la ayuda de JavaScript y que son soportadas para las plataformas de Android y iOS. Es similar al framework de ReactJS usado para desarrollo web, pero este usa como bloques de construcción componentes nativos en lugar de componentes web.

Para la programación en React Native se usan elementos denominados componentes, los cuales permiten separar la interfaz de usuario en piezas independientes, reutilizables y pensar en cada pieza de forma aislada. Estos componentes son equivalentes a funciones de JavaScript, aceptan parámetros de entrada llamados props y retornan elementos de React que describen lo que debe aparecer en pantalla. Así, una aplicación móvil desarrollada con React Native tendrá una serie de componentes que se mostrarán en pantalla y se actualizarán solamente cuando el estado de las variables (props) que manejan cambien y solamente aquellos componentes que lo requieran y no toda la aplicación, generando una mayor eficiencia en las respuestas ante los cambios realizados por el usuario.

4. Metodología

En esta sección se presentan los pasos seguidos para la realización del proyecto, desde la construcción de la base de datos para entrenar al modelo hasta la implementación de la aplicación móvil destinada al uso del modelo entrenado. En la **Tabla 1** se lista el proceso seguido para la realización.

Tabla 1. Metodología propuesta para los objetivos planteados.

Objetivo específico	Metodología
Construir una base de datos con suficientes muestras para entrenar el modelo de *Deep Learning*. Dicha base de datos se construirá tomando fotos de 5 tipos de productos, por lo que se tendrán 5 clases entre productos de panadería y bebidas.	Se estudió la base de datos existente con el fin de validar si era lo suficientemente representativa para el problema.
	Se tomaron y generaron nuevas imágenes para la base de datos para tener una cantidad considerable para el entrenamiento del modelo.
Realizar el preprocesado de las imágenes contenidas en la base de datos.	Se realizaron cambios de las dimensiones de las imágenes con el fin de facilitar el entrenamiento de la red.
	Se estudiaron y efectuaron transformaciones sobre las imágenes que permitieron brindar una mayor cantidad de muestras al sistema, tales como: suavizado, rotación, etc.
Hacer uso de un modelo previamente entrenado de Deep Learning como punto de partida para el entrenamiento del sistema (transferencia de aprendizaje), y evaluar su desempeño.	Se estudiaron diferentes arquitecturas de redes de Deep Learning y se seleccionó la que se consideró mejor.
	Se establecieron las condiciones de prueba para el escenario controlado. Dicho escenario hace referencia a imágenes destinadas para validación en donde se observan varios de los productos.

	Se realizó la evaluación de métricas que permitieron describir la precisión del sistema, tales como matriz de confusión, recall, etc.
	Se realizaron ajustes a la red para mejorar el desempeño del sistema al ser usado.
Realizar pruebas del sistema en un escenario real con el fin de evaluar su desempeño en dicho ambiente.	Selección y estudio de desarrollo de aplicaciones móviles para hacer la interfaz de usuario.
	Se desarrolló la interfaz de usuario para la plataforma de dispositivos móviles.
	Se realizaron pruebas para validar el correcto funcionamiento de la integración del sistema con la interfaz de usuario.
	Se realizaron ajustes en la integración para mejorar el desempeño en el uso del modelo.
	Se realizaron pruebas del sistema completo para comprobar el correcto funcionamiento tanto en el frontend (interfaz de usuario) como en el backend (servicio que ejecuta el modelo).

4.1. Base de datos

La base de datos se formó en un principio con el desplazamiento a un punto de venta para tomar 20 fotos de 5 productos de panadería: alfajor de maicena, alfajor de chocolate, parmesana, corazón y aromática. Las fotos fueron tomadas de forma individual hasta tener 20 por clase y, adicionalmente, se tomaron otras fotos de los productos agrupados para hacer pruebas del funcionamiento del sistema una vez entrenado.

La cantidad de fotos por clase es baja, pero se tuvo en cuenta el proceso de transferencia de aprendizaje que permite entrenar con muy pocos datos y aún así obtener buenos resultados haciendo uso de pesos previamente obtenidos y que proporcionan los desarrolladores del modelo para el modelo utilizado en el proyecto.

4.2. Pre-procesado de base de datos

Una vez obtenidas 20 fotos por cada producto, se aplicaron dos acciones sobre ellas: La primera consiste en llevar a cabo la técnica de data augmentation sobre las muestras existentes, esto se logra aplicando 4 transformaciones sobre las imágenes y almacenándolas para su posterior uso, como se observa en la **Figura 4**. Las transformaciones aplicadas a las imágenes son rotación, giro, filtro gaussiano y dilatación; generando así 100 imágenes en total por cada clase.

La segunda acción sobre la base de datos consiste en un cambio de dimensiones para hacer más fácil la manipulación de las imágenes. Para esto se hace uso de una función de la librería de OpenCV para redimensionar imágenes y se establece el tamaño de las estas en 256x256 píxeles, por ser un valor típico en las implementaciones de sistemas con CNN. En la **Figura 5** se observan varias fotos con las modificaciones mencionadas.

```
def data_augmentation(img_loc, img_rsz):
    '''
        Parámetros
            img_loc: Ubicación de la imagen
            img_rsz: Tamaño para las imágenes generadas

        Retorno
            Ninguno
    '''

    dim = img_rsz
    img = cv2.imread(img_loc)
    img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

    db_path    = img_loc.split('/')[0]
    img_name   = img_loc.split('/')[1].split('.')[0]
    img_format = img_loc.split('/')[1].split('.')[1]

    rotated_image = rotate_image(img)
    flipped_image = flip_image(img, axis=-1)
    gb_image = gaussian_blur(img)
    #rotated_gb_image = rotate_image(gb_image)
    dilated_image = dilate_image(img)

    cv2.imwrite(db_path + '/' + img_name + '.' + img_format, img)
    cv2.imwrite(db_path + '/' + img_name + '_r.' + img_format, rotated_image)
    cv2.imwrite(db_path + '/' + img_name + '_f.' + img_format, flipped_image)
    cv2.imwrite(db_path + '/' + img_name + '_gb.' + img_format, gb_image)
    cv2.imwrite(db_path + '/' + img_name + '_d.' + img_format, dilated_image)
```

Figura 4. Función implementada para redimensionar las imágenes, aplicar data augmentation y guardar su resultado.

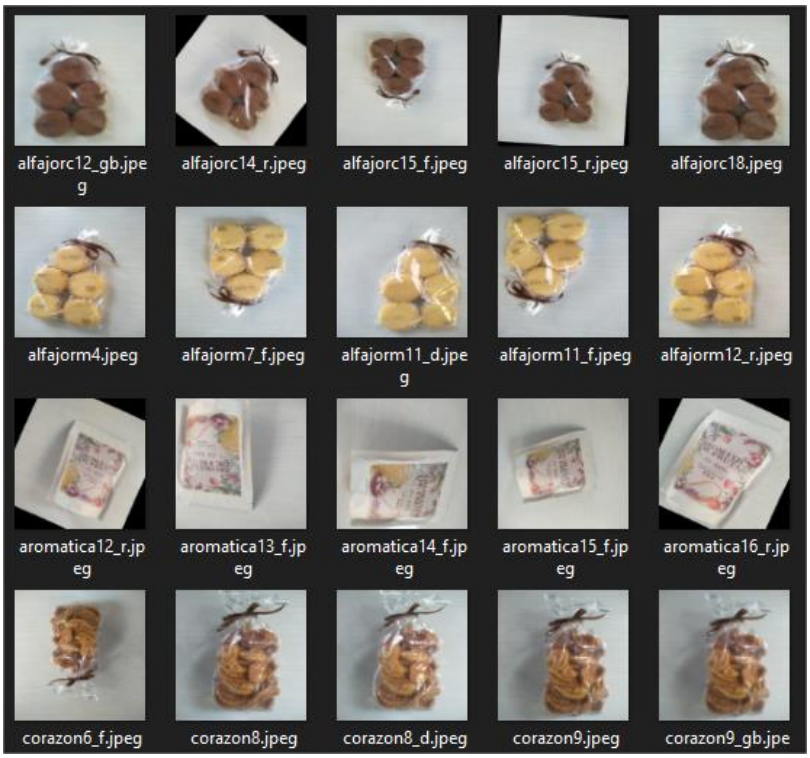

Figura 5. Ejemplo de imágenes modificadas del dataset.

4.3. Modelo Deep Learning

Una vez realizado el procesamiento de la base de datos, el paso siguiente consiste en realizar el etiquetado de cada una de las imágenes haciendo uso del programa LabelImg, el cual genera un archivo con extensión .txt con las coordenadas del rectángulo delimitador donde se encuentra el producto y la clase a la que este pertenece. Este es el proceso más arduo del proyecto ya que requiere que se recorra cada una de las imágenes y manualmente se asigne el rectángulo delimitador para generar el archivo requerido. En la **Figura 6** se muestra el proceso de etiquetado para una imagen del conjunto de entrenamiento usando la herramienta y la salida generada.

Con el dataset correctamente creado, se crea la estructura de carpetas utilizada para el proyecto, la cual incluye las imágenes y las etiquetas con sus respectivos subdirectorios para los conjuntos de entrenamiento, prueba y validación. Adicional a esto se incluye un archivo de configuración con extensión .yaml, el cual brinda información a la red de YOLOv5 al momento de entrenar, incluyendo las rutas de los diferentes

conjuntos, el número de clases y los nombres de las clases, tal como se observa en la **Figura 7**.

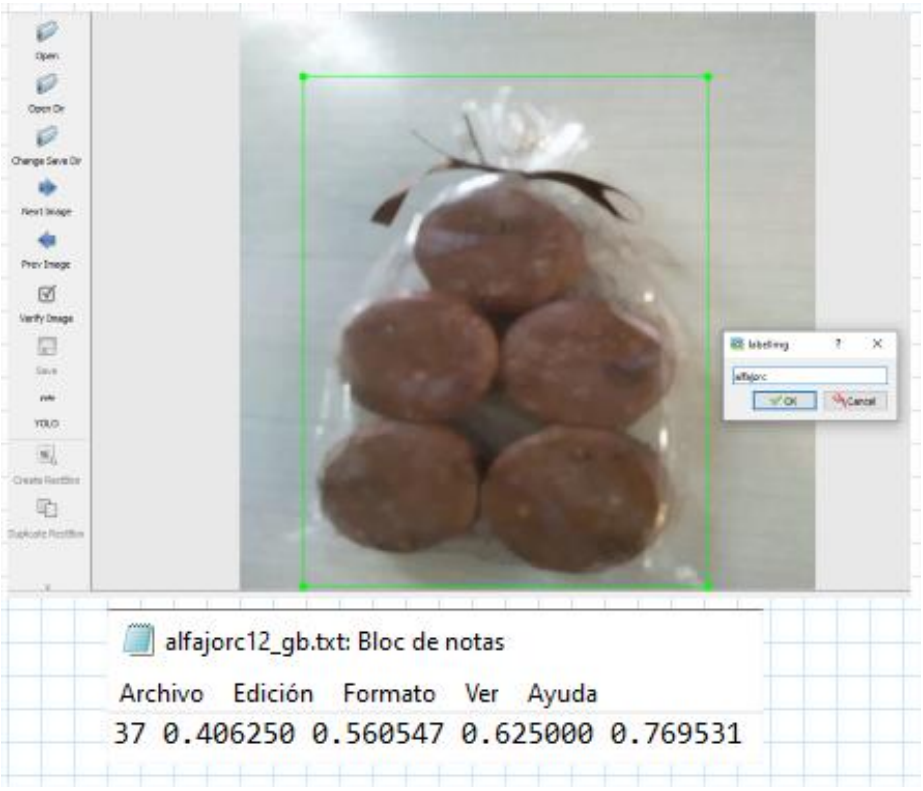

Figura 6. Arriba: Etiquetado de una de las imágenes del conjunto de entrenamiento usando la herramienta LabelImg. Abajo: Archivo de salida del proceso de etiquetado.

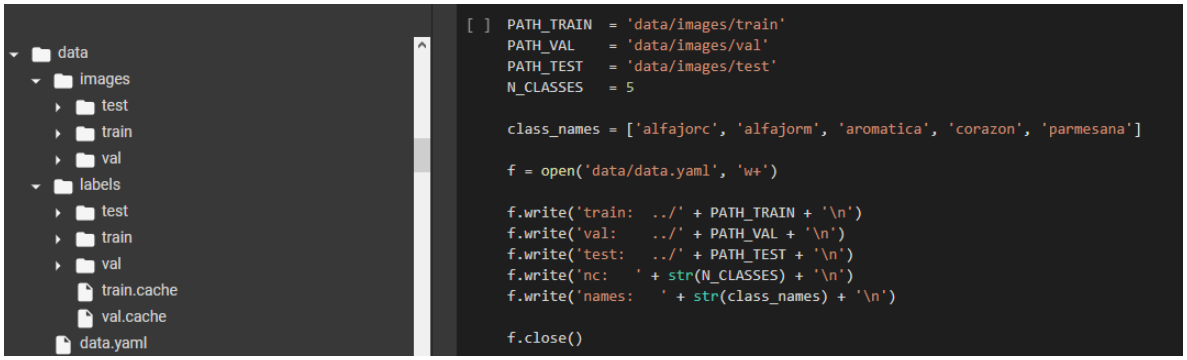

Figura 7. Izquierda: Estructura de archivos generada. Derecha: Generación de archivo de configuración para YOLOv5.

Posterior a la organización de los archivos necesarios, se realiza el entrenamiento de la red usando como pesos iniciales los que tiene configurados por defecto el proyecto de YOLOv5 clonado desde el repositorio de github de ultralytics, y la configuración establecida en el archivo .yaml. Las primeras pruebas se realizaron con el conjunto de validación utilizando el modelo entrenado con la transferencia de aprendizaje, dando resultados satisfactorios.

4.4. Aplicación móvil

Para la interfaz de usuario se decidió desarrollar una aplicación móvil, puesto que es más fácil para el usuario tomar la foto desde el celular y que este realice el proceso de clasificación y conteo. En cuanto al framework se elige React Native por su similitud con ReactJS, con el cual se tiene experiencia previa; además porque permite desarrollar una aplicación basada en componentes que se puede desplegar tanto para Android como para iOS.

Para el frontend (interfaz de usuario) se diseñaron dos vistas, la primera corresponde al inicio compuesto del logo de la empresa CincoE, el nombre del proyecto y un botón para acceder a la siguiente vista, en la que se encuentran las opciones de tomar foto o subir una existente en el celular para ser analizada por el modelo de YOLOv5, estas vistas se observan en la **Figura 8**.

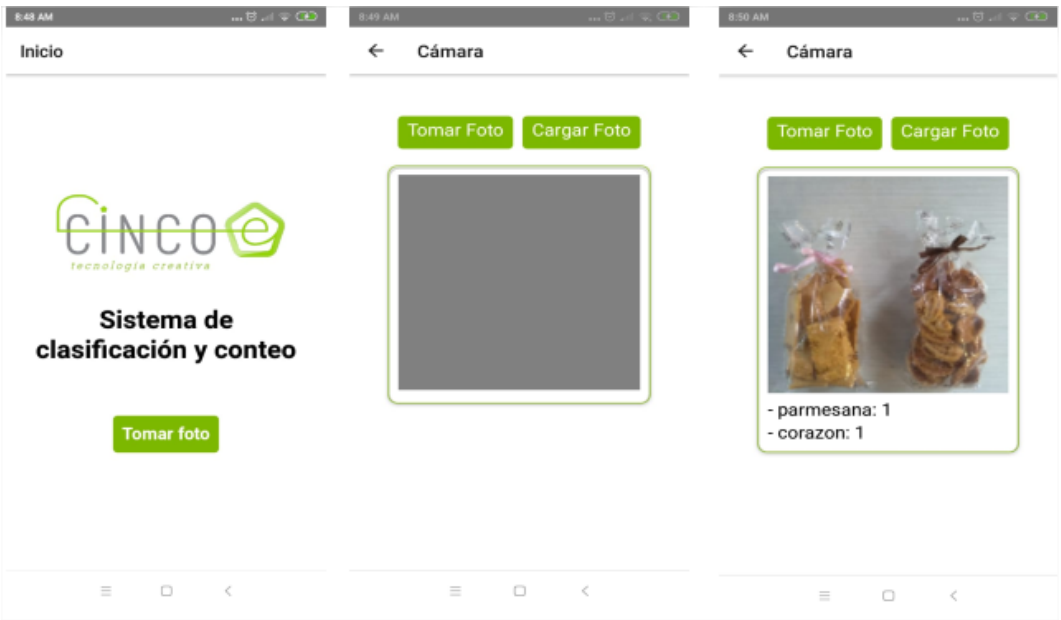

Figura 8. Izquierda: Vista de inicio. Medio: Vista de procesamiento sin haber elegido imagen. Derecha: Vista de procesamiento al elegir imagen.

Al seleccionar alguna de las dos opciones, la imagen es enviada a un servicio de backend implementado con la librería de Python fastAPI. Este es un servicio del tipo POST que recibe como parámetro un archivo, que para el sistema sería una imagen, la preprocesa usando la librería de OpenCV y la guarda temporalmente para usar el modelo de YOLOv5 y poder retornar al frontend una respuesta en formato JSON que contiene todas las detecciones encontradas usando el modelo entrenado. En la **Figura 9** se puede ver el código que realiza lo descrito.

```python
@app.post("/upload_photo")
async def upload_photo(file: UploadFile = File(...)):
    print("Starting...")
    content = await file.read()
    nparr = np.fromstring(content, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    resized_image = cv2.resize(img, (256, 256), interpolation = cv2.INTER_AREA)
    cv2.imwrite('image.jpg', resized_image)

    TEST_WEIGHTS = 'best.pt'
    IMAGE_PATH   =  'image.jpg'

    # Model
    model = torch.hub.load('ultralytics/yolov5', 'custom', path=TEST_WEIGHTS)

    # Inference
    results = model(IMAGE_PATH, size=256)
    results.print()
    print(results.pandas().xyxy[0].to_json())

    os.remove(IMAGE_PATH)

    return { "result": results.pandas().xyxy[0].to_json() }
```

Figura 9. Código que representa el servicio POST usado para clasificar una imagen enviada desde el frontend**.**

Con el modelo de YOLOv5, el frontend desarrollado con React Native y el backend desarrollado con FastAPI, se completa el esquema del sistema de clasificación y conteo. La estructura mostrada en la **Figura 10** representa los 3 módulos del proyecto y sus respectivas conexiones para el intercambio de información.

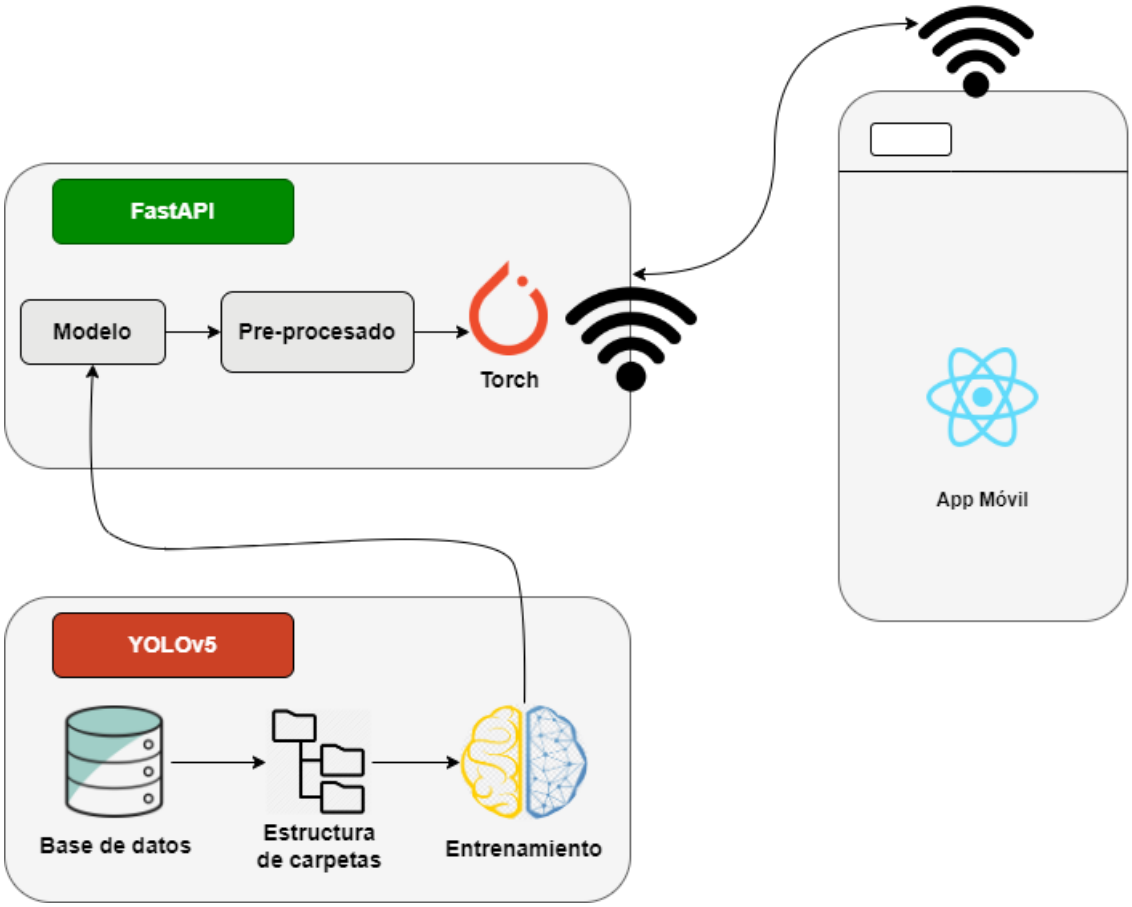

Figura 10. Estructura del proyecto.

5. Resultados y análisis

Para el modelo entrenado con YOLOv5 se tiene la matriz de confusión mostrada en la **Figura 11**, en donde se observa que todas las clasificaciones de los productos en alguna de las clases son correctas. Para las pruebas del modelo se usaron 50 imágenes almacenadas en el directorio de validaciones y otras 16 tomadas después, y se puede observar que efectivamente se obtienen buenos resultados para la clasificación individual de los productos en la **Figura 12**, en donde se ven varias de las imágenes de los productos de interés con distintas transformaciones y con sus respectivas probabilidades de pertenencia a la clase que aparece escrita en el rectángulo delimitador, validando la información mostrada por la matriz de confusión.

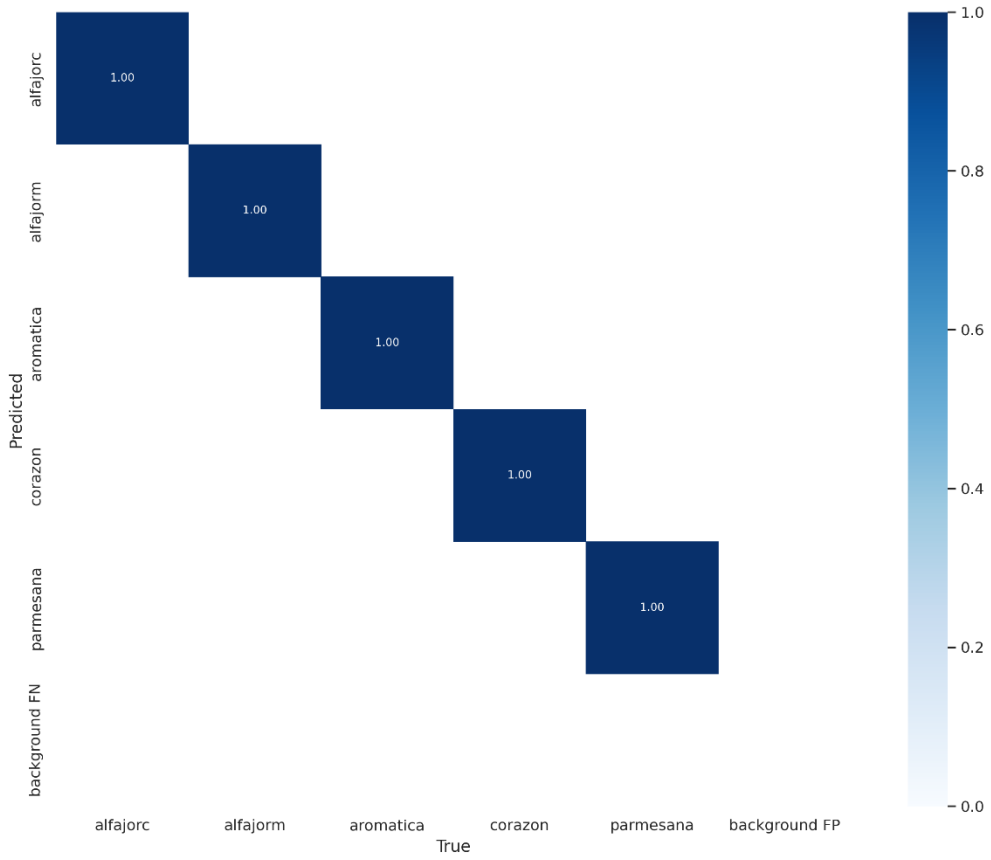

Figura 11. Matriz de confusión entregada por YOLOv5.

Figura 12. Múltiples detecciones realizadas por el modelo entrenado de YOLOv5.

Sin embargo, al realizar más pruebas se observa que para algunas de las imágenes se presenta error, en especial para aquellas con similitudes que se encuentran juntas, como es el caso del alfajor de maicena y el alfajor de chocolate, situación que se evidencia en la **Figura 13**. Aún así, se tiene que para la gran mayoría de las imágenes probadas los resultados son satisfactorios, incluso para imágenes borrosas como la mostrada en la **Figura 14**; aunque se podría robustecer el sistema al incluir en el entrenamiento muchas más imágenes que contengan combinaciones de los productos de tal forma que la CNN pueda distinguir los productos cuando estén agrupados.

Figura 13. Izquierda: Imagen con todas las detecciones correctas. Derecha: Imagen con detección incorrecta.

Figura 14. Imagen borrosa con productos detectados correctamente.

Para la aplicación móvil se obtenía en un principio un desempeño deficiente puesto que se transmitían las imágenes sin cambiar las dimensiones, por lo que el sistema no funcionaba bien. Para solucionar esto se implementó el preprocesado de la imagen en el backend, lo cual hizo que se igualara el desempeño de la aplicación con el del modelo probado individualmente, como se logra apreciar en la **Figura 15**,

en donde se tomó una foto del monitor de un computador con la imagen (generando una imagen con pésima calidad) y aún así se lograron detectar ambos productos.

Figura 15. Detección y conteo de productos desde la aplicación móvil.

6. Conclusiones

- El preprocesado de las imágenes jugó un papel importante, ya que permitió aumentar la base de datos aplicando transformaciones sobre las imágenes existentes, además de adecuarlas para el uso del modelo. Se puede pensar en robustecer la base de datos aplicando aún más transformaciones con el fin de obtener un mejor desempeño.
- La construcción de la base de datos para el entrenamiento del modelo fue la etapa más engorrosa del proyecto, ya que implica tomar cada una de las imágenes y etiquetarlas.
- Se pudo observar la gran utilidad del algoritmo YOLOv5 para la detección de objetos en imágenes. También se notó la importancia del uso de la transferencia de aprendizaje al usar los pesos predefinidos en el modelos, permitiendo entrenar la red con mucho menos imágenes de las que se requerirían sin este método.
- De los experimentos realizados se pudo observar la gran eficiencia que logró el sistema con pocos datos de entrenamiento al igual que su correcta integración con la aplicación móvil desarrollada, permitiendo la versatilidad de tomar fotos desde el dispositivo o usar una imagen almacenada en este. Sin embargo, se puede mejorar el desempeño del sistema para disminuir las detecciones erróneas usando posteriormente más imágenes con varios productos e imágenes que no contengan ningún producto para el entrenamiento del modelo de YOLOv5.

7. Referencias bibliográficas

A. Takimoglu. (30 de abril de 2021). What is Data Augmentation? Techniques & Examples in 2022 [online]. Disponible en: https://research.aimultiple.com/data-augmentation/.

Ahmad, Jamil & Muhammad, Khan & Baik, Sung. (2017). Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search. PLOS ONE. 12. e0183838. 10.1371/journal.pone.0183838.

Artamonov, N & Yakimov, Pavel. (2018). Towards Real-Time Traffic Sign Recognition via YOLO on a Mobile GPU. Journal of Physics: Conference Series. 1096. 012086. 10.1088/1742-6596/1096/1/012086.

AWS. (12 de octubre de 2021). Machine Learning Lens [online]. Disponible en: https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/data-preprocessing.html.

AWS. (21 de mayo de 2021). ¿Qué es una red neuronal? [online]. Disponible en: https://aws.amazon.com/es/what-is/neural-network/.

Cong, An & Tran, Nghi & Duong Trung, Nghia. (2020). “Recognition and Quantity Estimation of Pastry Images Using Pre-training Deep Convolutional Networks”. 10.1007/978-981-33-4370-2_15.

FastAPI. FastAPI [online]. Disponible en: https://fastapi.tiangolo.com.

Hassan, Hassan & Negm, Abdelazim & Zahran, Mohamed & Saavedra, Oliver. (2015). ASSESSMENT OF ARTIFICIAL NEURAL NETWORK FOR BATHYMETRY ESTIMATION USING HIGH RESOLUTION SATELLITE IMAGERY IN SHALLOW LAKES: CASE STUDY EL BURULLUS LAKE: International Water Technology Journal.

IBM. (15 de julio de 2020). Machine Learning [online]. Disponible en: https://www.ibm.com/co-es/cloud/learn/machine-learning.

IBM. (28 de abril de 2021). Augmenting the data set [online]. Disponible en: https://www.ibm.com/docs/en/maximo-vi/1.3.0?topic=model-augmenting-data-set.

KRIEGESKORTE, Nikolaus; GOLAN, Tal. “Neural network models and deep learning”. Current Biology, 2019, vol. 29, no 7, p. R231-R236.

Numpy. What is NumPy? [online]. Disponible en: https://www.python.org/doc/essays/blurb/.

OpenCV. About [online]. Disponible en: https://opencv.org/about/.

Python. What is Python? Execute Summary [online]. Disponible en: https://numpy.org/doc/stable/user/whatisnumpy.html.

RAJ, Rishi, et al. Crossover based technique for data augmentation. Computer Methods and Programs in Biomedicine, 2022, p. 106716.

React Native. (30 de marzo de 2022). Learn the Basics. Disponible en: https://reactnative.dev/docs/tutorial.

REDMON, Joseph, et al. You only look once: Unified, real-time object detection. En Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 779-788

S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

S. W. Chen et al., "Counting Apples and Oranges With Deep Learning: A Data-Driven Approach," in IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 781-788, April 2017, doi: 10.1109/LRA.2017.2651944.

Sammut C., Webb G.I (2017) Data Augmentation. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA.

Ultralytics. (18 de mayo de 2020). YOLOv5 Documentation [online]. Disponible en: https://docs.ultralytics.com.

Yuchen Wei, Son Tran, Shuxiang Xu, Byeong Kang, Matthew Springer, "Deep Learning for Retail Product Recognition: Challenges and Techniques", Computational Intelligence and Neuroscience, vol. 2020, Article ID 8875910, 23 pages, 2020.

ZHU, Lili, et al. “Deep learning and machine vision for food processing: A survey”. Current Research in Food Science, 2021, vol. 4, p. 233-249.