UNIVERSIDAD MAYOR REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE CIENCIAS Y TECNOLOGÍA

CARRERA DE ING. EN CIENCIAS DE LA COMPUTACION.

PROYECTO DE SIS-330

"SOFTWARE DE AYUDA PARA DETECTAR Y CLASIFICAR EL PESCADO EN BUEN Y MAL ESTADO"

Universitario: Chojllu Coa Wilson

Docente: Pacheco Lora Carlos Walter

Sucre-Bolivia

Índice de Contenido

1. RESUMEN

El estudio tuvo como objetivo identificar regiones sospechas del pescado en mal estado y clasificar por tres tipos de pescado las cuales son Pacú, Sábalo, y Surubí. Como resultado, llevamos a cabo un estudio de caso para la detección de pescados, se creó un conjunto de datos de más de 15000 imágenes de pescados. Las imágenes se obtuvieron sacando fotos con una cámara de celular (235 imágenes) así como de Internet (800 imágenes). Usando un aumento de datos se logró obtener (más de 15000 imágenes), conjunto de datos se entrenó utilizando el algoritmo YOLOv5 pre entrenado. El algoritmo YOLOv5 se utilizó por primera vez en este estudio para detectar regiones en mal estado de los pescados en su superficie.

2. ANTECEDENTES

En la actualidad el consumo del pescado en Bolivia es consumido por su gran valor nutricional, además es muy variado por la gran cantidad que se puede encontrar en los mercados. Entre los pescados más consumidos son:

- Sábalo.
- Sábalo Dorado de escamas.
- Pirahiba.
- Yatorana.
- Surubí
- Chipi chipi.
- Pacú

3. SITUACION PROBLEMATICA

Actualmente la manera en la cual se determina el estado, tipo de pescado es basándose en la capacidad de visión, lo que a la larga genera muchos problemas en cuanto a la confianza del análisis realizado, ya que no todos poseen una misma idea sobre cuando un pescado deja de estar en buen estado o al saber de qué tipo es.

En ocasiones podemos ser engañados por las vendedoras ofreciéndonos un tipo de pescado que luego de ser comprado y constatarlos no sea el mismo que nos ofreció.

En muchos casos podemos comprar pescado en mal estado que nos puede generar malestares, intoxicación después de consumirlos.

El pescado en mal estado es peligroso para la salud humana y una seria amenaza para comercializadores de este producto. El mal estado de los pescados provoca importantes pérdidas económicas, así como enfermedades de **escombroides**, donde los principales síntomas son:

- Cólicos abdominales
- Diarrea (intensa y acuosa)
- Problemas respiratorios, incluso sibilancias y presión en el pecho (en casos graves)
- Piel de la cara y el cuerpo extremadamente roja

- Sofoco
- Picazón y ronchas
- Náuseas y vómitos
- Sabor picante o amargo

(EC., 2019)

4. PROBLEMA PRINCIPAL

El consumo del pescado en mal estado es peligroso para la salud humana que nos puede ocasionar muchas enfermedades.

5. OBJETIVOS

5.1 Objetivo General

Desarrollar una aplicación de detección y clasificador de imágenes de los pescados en mal estado. Para ayudar a la población en general que la necesite.

5.2 Objetivo Específicos

- Elaborar un modelo para la detección y clasificación de pescados en mal estado.
- Construir o conformar un conjunto de imágenes (Dataset) que permitan la construcción de un modelo de detección de pescados en mal estado y clasificarlos.
- Mediante el uso de reconocimiento de imágenes identificar y clasificar un pescado con un porcentaje de precisión igual o mayor al 90%.
- Mediante el uso de reconocimiento de imágenes identificar el estado de un pescado con un porcentaje de precisión igual o mayor al 90%.
- Desarrollar e implementar una aplicación móvil que muestre los resultados de identificación de pescados con el fin de validar su desempeño

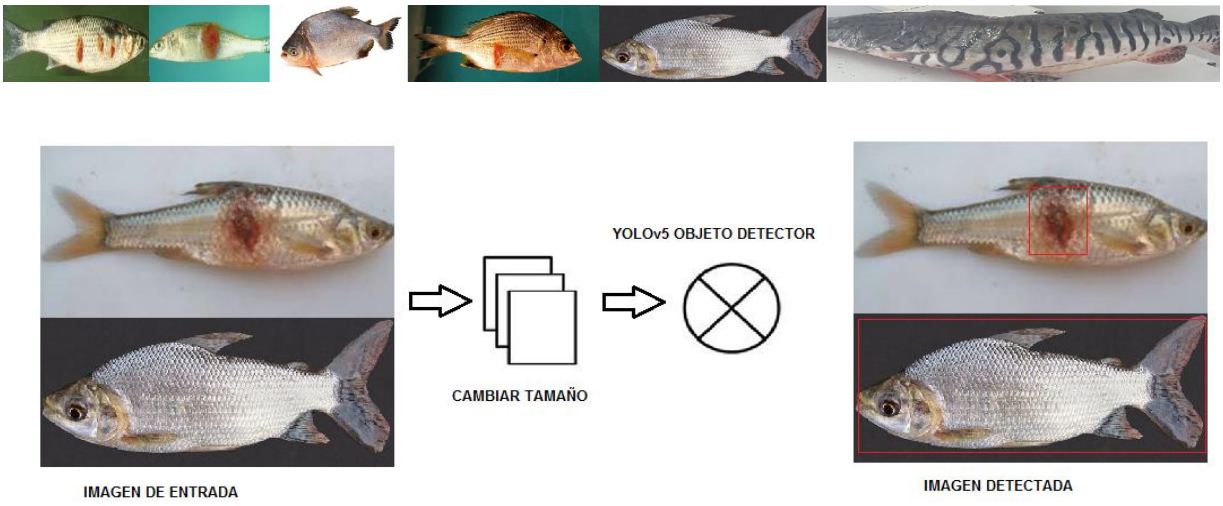
6. FUNDAMENTOS TEORICOS

La detección de objetos es una rama importante en el campo de la visión artificial y el procesamiento de imágenes. La detección de objetos es el proceso de identificar ocurrencias de un tipo específico de objeto en imágenes y videos. Los algoritmos de detección de objetos han recibido mucha atención en el aprendizaje profundo **(deep learning)** (Yan P. y., 2020). Los algoritmos de detección de objetos basados en el aprendizaje profundo han avanzado rápidamente en los últimos años. Podeos mencionar YOLO un algoritmo de muy alta precisión y velocidad se ha utilizado en varias tareas de detección de escenas (Kuznetsova A, 2020). Al mismo tiempo, el sistema YOLO calcula todas las características de la imagen y predice todos los objetos. YOLOv5 es la quinta generación de YOLO, escrito en lenguaje de programación Python (Thuan, 2021). Según varios estudios, YOLOv5 supera al resto del modelo YOLO en términos de precisión y velocidad. En algunos estudios recientes, YOLOv5 se utilizó para detectar varios objetos. El modelo YOLOv5 fue utilizado por (Yan, 2021)**.** Para detectar manzanas en huertos mediante robots recolectores. En ambos estudios, la velocidad de detección y la precisión fueron notables en comparación con otros modelos de YOLO. En otro estudio, (E. Cengil, 2021)**.** Utilizaron un algoritmo YOLOv5 previamente entrenado y un conjunto de datos de ocho especies de hongos venenosos para detectar la detección de hongos venenosos. Como resultado, decidí transmitir la investigación actual utilizando YOLOv5. Sin embargo, hasta donde sé, ningún estudio ha utilizado YOLOv5 para detectar pescado en mal estado y clasificarlos por su tipo.

7. COMPONENTE IA

7.1 Descripción y esquema del modelo

La figura muestra el esquema general de los modelos propuestos para la detección y clasificación de pescados en mal estado. El modelo pre entrenado están compuesta por una arquitectura basada en YOLOv5.



El primer modelo consiste en evaluar la capacidad de localizar regiones sospechosas de los pescados y El segundo modelo consiste en clasificar el tipo de pescado.

Se entrenara mediante Google Colab, Utilizamos un cuaderno desarrollado por Roboflow. ai (Roboflow, 2016). Que se basa en YOLOv5 y utiliza pesos COCO preentrenados. Se eligió un número adecuado de épocas para entrenar un conjunto de datos de pescados. Para entrenar el modelo se seleccionó 150 épocas lo cual tomó aproximadamente 2 horas. Por cada modelo, En el directorio YOLOv5 hay un archivo 'train.py' para entrenar el modelo YOLOv5. Usando el comando bash '!python train. py <parámetros>' el modelo fue entrenado en consecuencia.

Los detalles del entrenamiento del modelo YOLOv5 son los siguientes.

- Tamaño de la imagen: 640
- Tamaño del lote: 10

- Descripción de datos: datos. Yaml
- Modelo Yolo: YOLOv5s.yaml
- Imagen: alto y ancho de las imágenes.
- Lote: tamaño del mini lote de imágenes para alimentar en una iteración.
- Épocas: el número de iteraciones de entrenamiento.
- Data: (tren, validación) directorio de datos y número de clase y nombre de clase se describió en este archivo YAML.
- cfg: los modelos se describen en los archivos de configuración del modelo YAML en el directorio 'modelo'. Hay cuatro versiones del modelo de diferentes tamaños. Se ha utilizado 'YOLOv5s.yaml' para entrenar.

7.2 Descripción de adquisición de datos y pre procesamiento realizado

Hay dos tipos de archivos de datos en el conjunto de datos.

A) Imágenes digitales sin procesar que contienen un total de más de 15000 imágenes con una gran diversidad de poses, ángulos, condiciones de iluminación, condiciones climáticas y fondos. Las imágenes están todas en formato JPG.

B) Archivos de anotaciones de imágenes que contienen más de 15000 imágenes. Estos archivos especifican las ubicaciones precisas de los objetos con etiquetas en las imágenes correspondientes. La anotación se realizó manualmente y los valores anotados también se guardaron en archivos txt. La carpeta se dividió en tres subcarpetas: **train, val y test**. Con la siguiente separación 87% para el entrenamiento, 8% para la validación y un 5% para el test. Por otro lado, la preparación del conjunto de datos consta de 3 pasos:

Recopilación de datos

Se recogieron imágenes de pescados de tres tipos de especies las cuales son: SABALO, SURUBI, PACU Y PESCADOS EN MAL ESTADO. Se recolectaron 235 imágenes. Las cuales fueron capturadas usando una cámara de celular Samsung A30. Las imágenes restantes 800 se obtuvieron de varias fuentes de Internet. En un total de 1035 imágenes.

Procesamiento y Anotación de datos

Después de tomar de los pescados de los diferentes tipos y pescados en mal estado, todas las imágenes se convirtieron al formato JPG. La herramienta de anotación que se uso es Roboflow, donde se utilizó para etiquetar cuidadosamente las imágenes. Cada imagen se abre en esta herramienta una a la vez. Luego, se dibujó manualmente una forma rectangular en el límite de un objeto para especificar su ubicación exacta en esa imagen por x1, y1, x2, y2. Finalmente, a cada objeto se le ha asignado una etiqueta, como 'SABALO' o 'SURUBI', etc. En LabelImg, los valores anotados se guardaron como archivos txt en formato YOLOv5.

Aumento de datos

Después del etiquetado, se utilizó la herramienta de Roboflow y el paquete de CLODSA para aumentarlas. El aumento de datos se realizó para aumentar la cantidad y diversidad de datos. Para generar nuevas imágenes a partir del conjunto de datos, se utilizaron algunas técnicas de aumento de datos, como voltear, recortar y transformar el espacio de color. De las 1035 imágenes se obtuvieron más de 15000 imágenes.

7.3 Técnicas y métricas de entrenamiento, evaluación y validación.

Curva P-R (Precision-Recall). La precisión (Pres) es la fracción de casos relevantes entre los casos recuperados. *El recall (Rec)* es la fracción de casos relevantes que se han recuperado sobre la cantidad total de casos relevantes. Las ecuaciones para estos casos son las siguientes:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

Para ejecutar las pruebas se utiliza la siguiente línea de código:

```
!python val.py --weights runs/train/exp/weights/best.pt --
data {dataset.location}/data.yaml --img 416 --iou 0.25 --half
```

Para el modelo de clasificación de pescados se obtuvieron las siguientes estadísticas

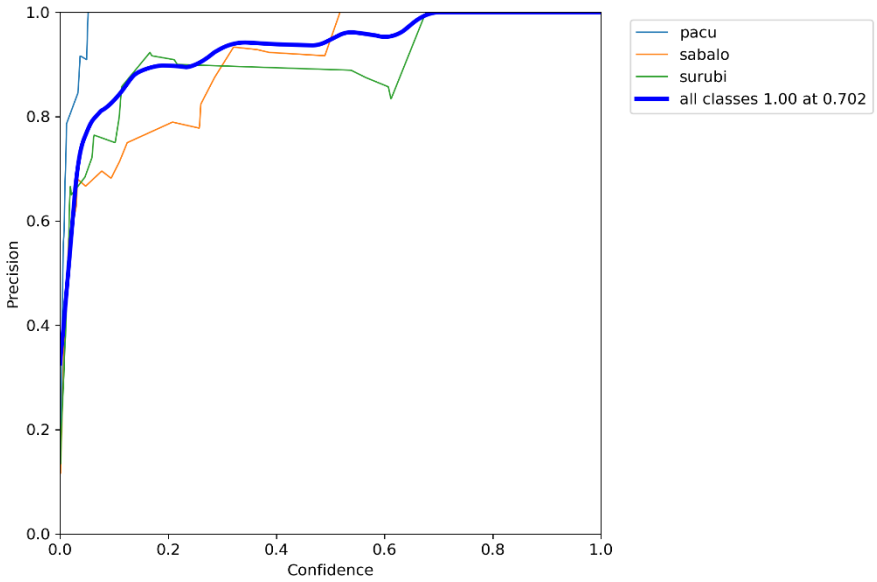


Ilustración 1: Precisión

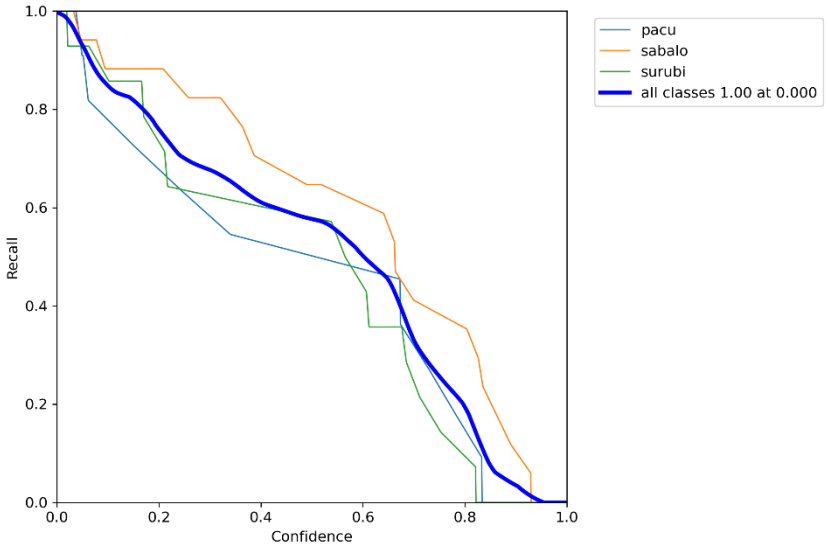


Ilustración 2: Recall

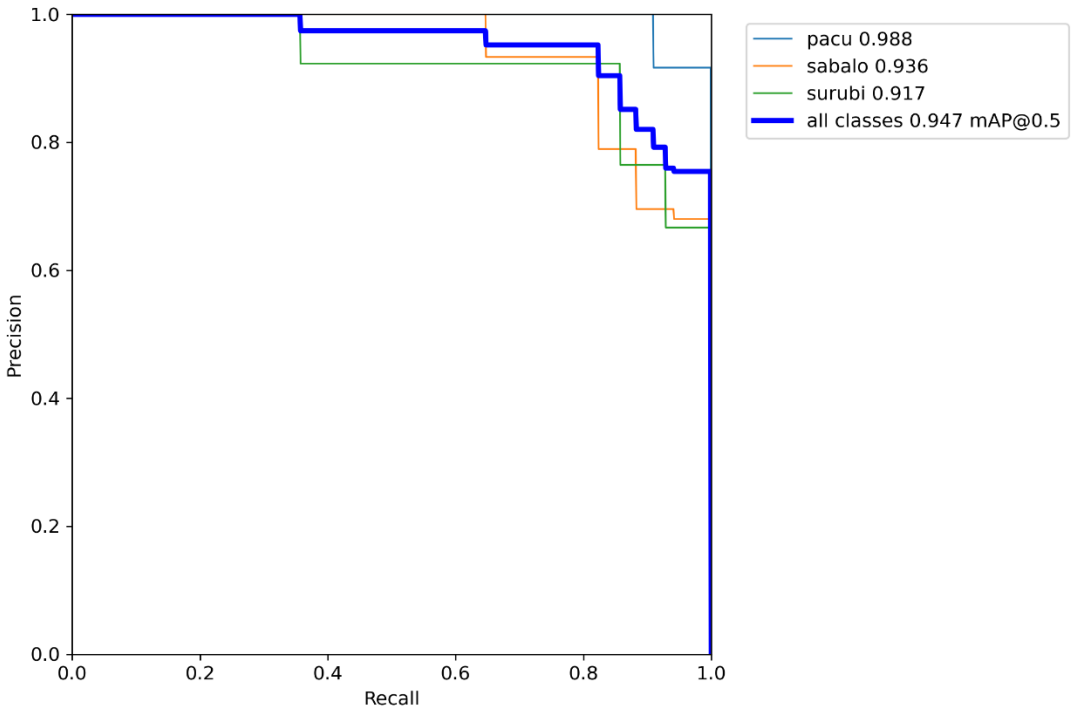


Ilustración 3: Precision-Recall

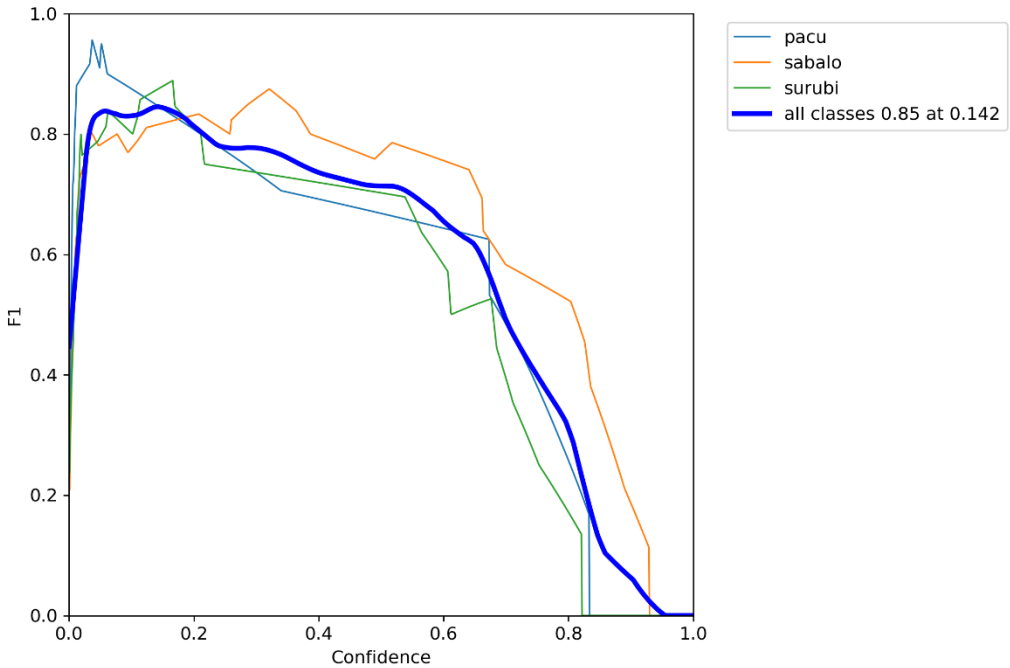


Ilustración 4: F1

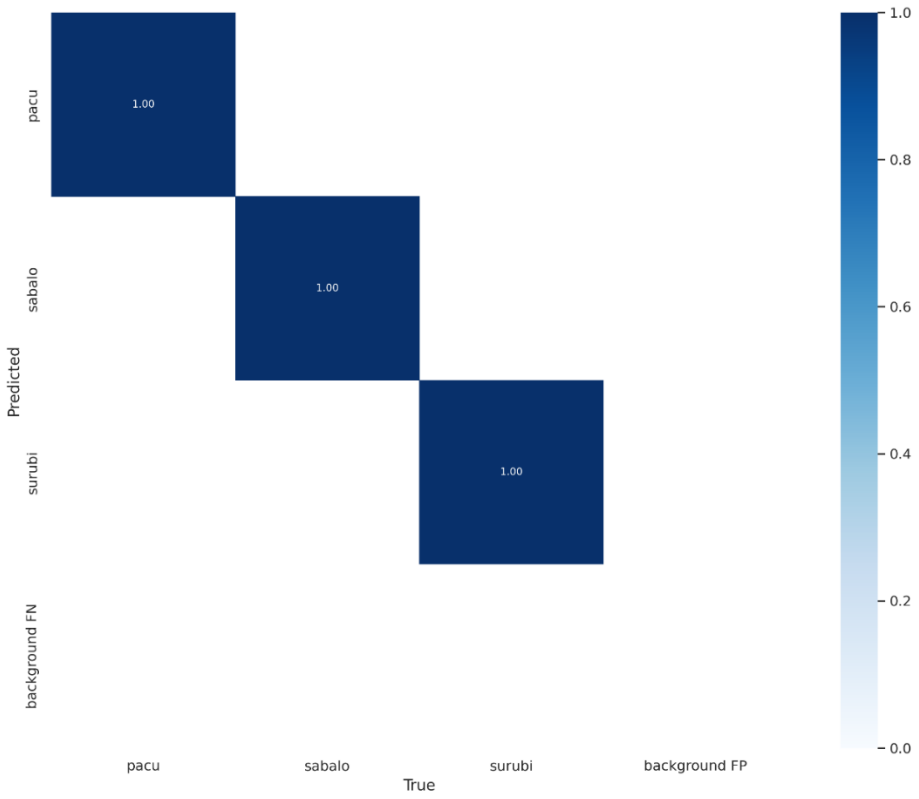


Ilustración 5: Matriz de confusión

Para el modelo identificación de pescados en mal estado se obtuvieron las siguientes estadísticas

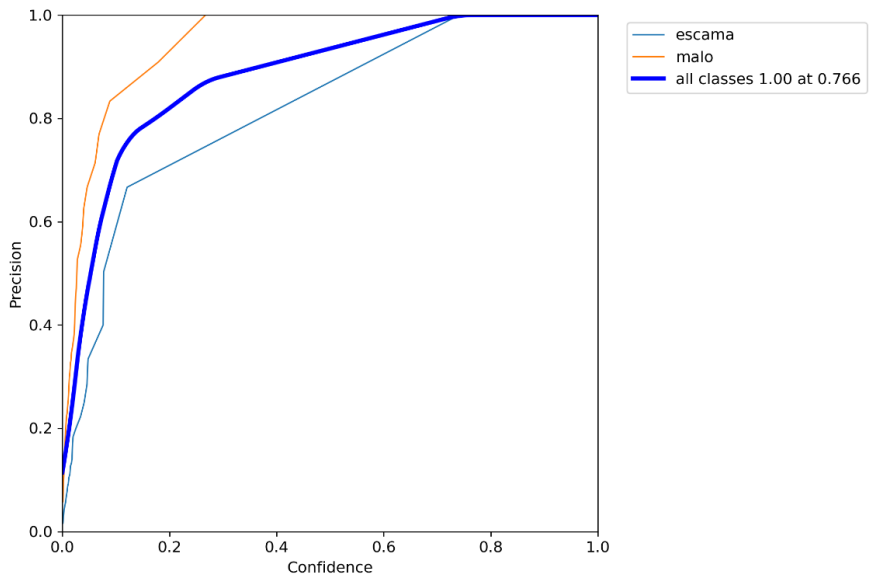


Ilustración 6: Precisión

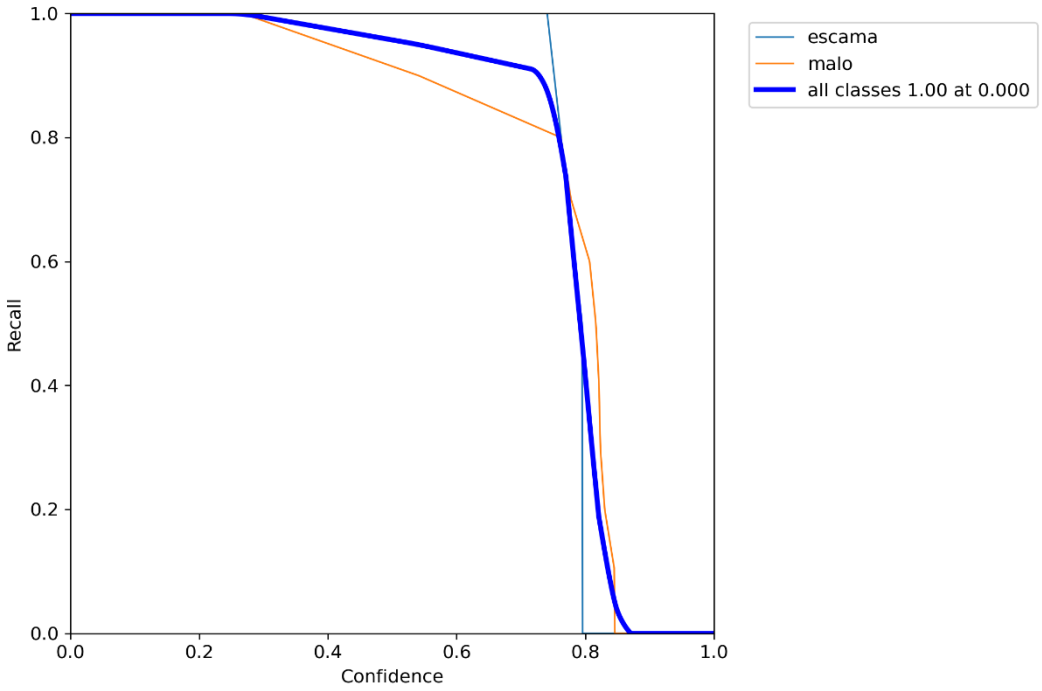


Ilustración 7: Recall

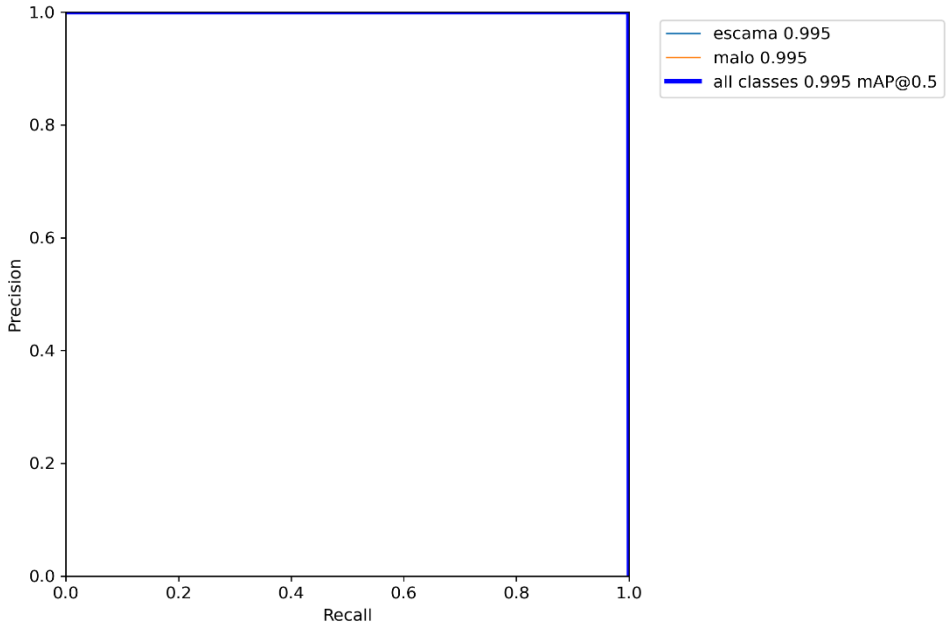


Ilustración 8: Precision-Recall

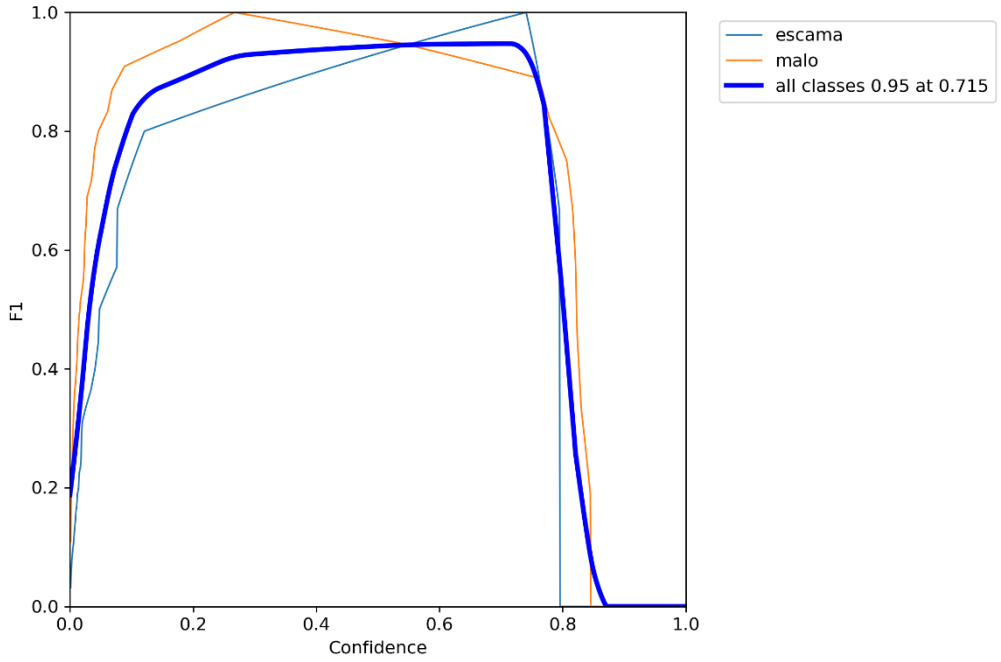


Ilustración 9: F1

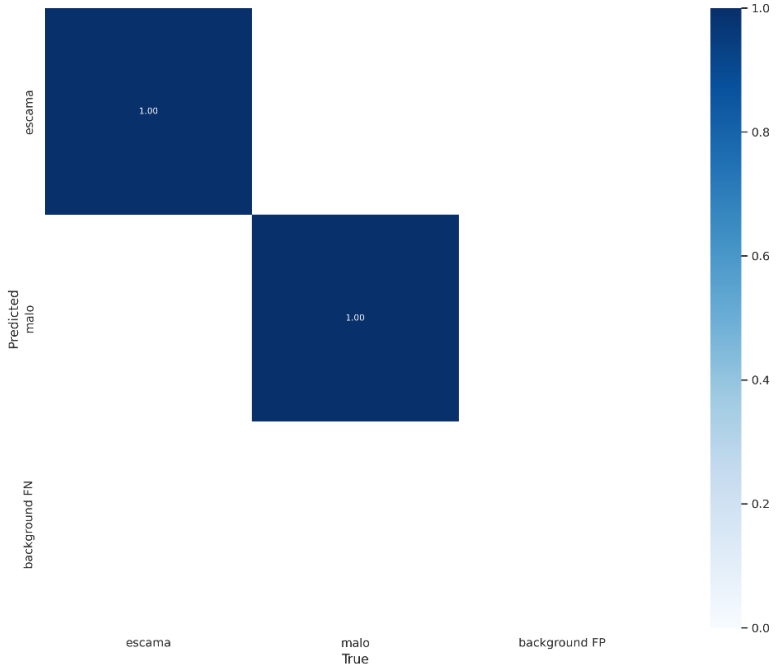


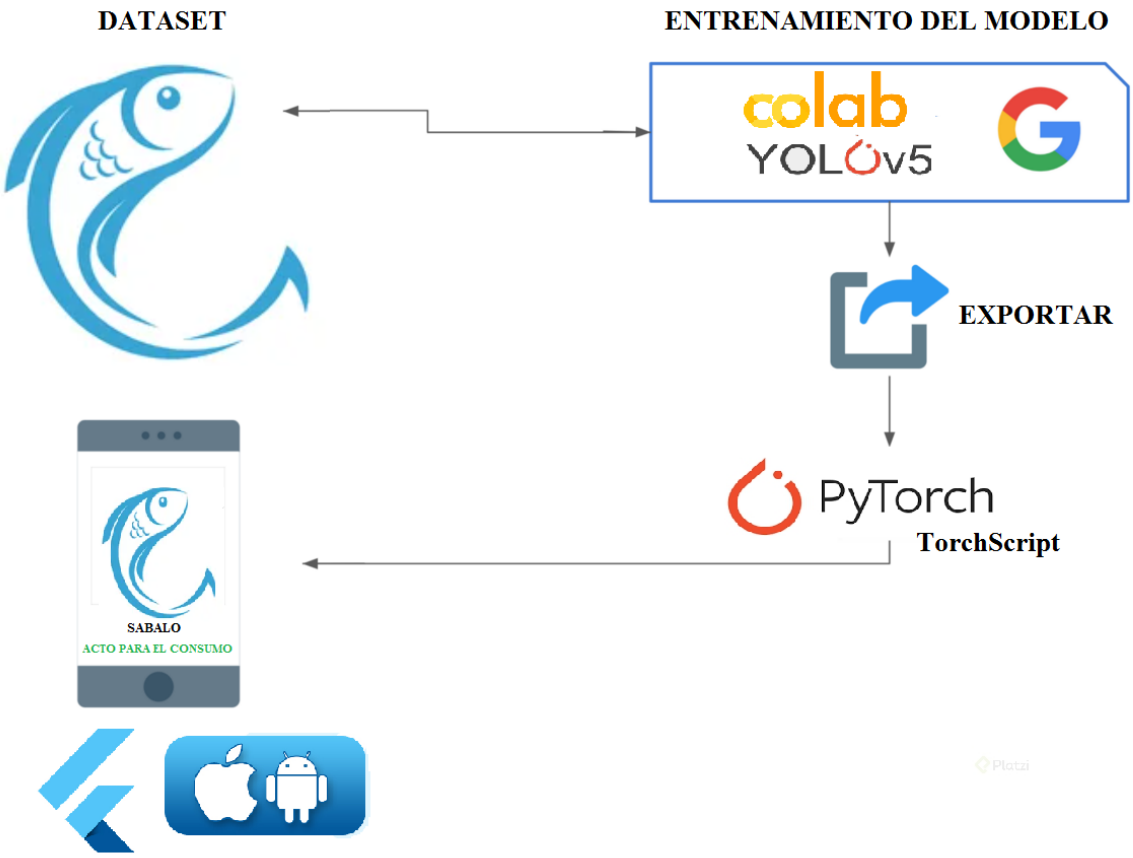
Ilustración 10: Matriz de confusión

7.4 Técnicas de Depuración Aplicadas.

8. DESCRIPCION DE TRABAJO REALIZADO

8.1 Esquema y descripción de Arquitecturas, Frameworks y componentes del Software Desarrollados y/o Empleados

La figura muestra el esquema para la detección y clasificación de pescados en mal estado.



Después de entrenamiento se procedió a los siguientes pasos:

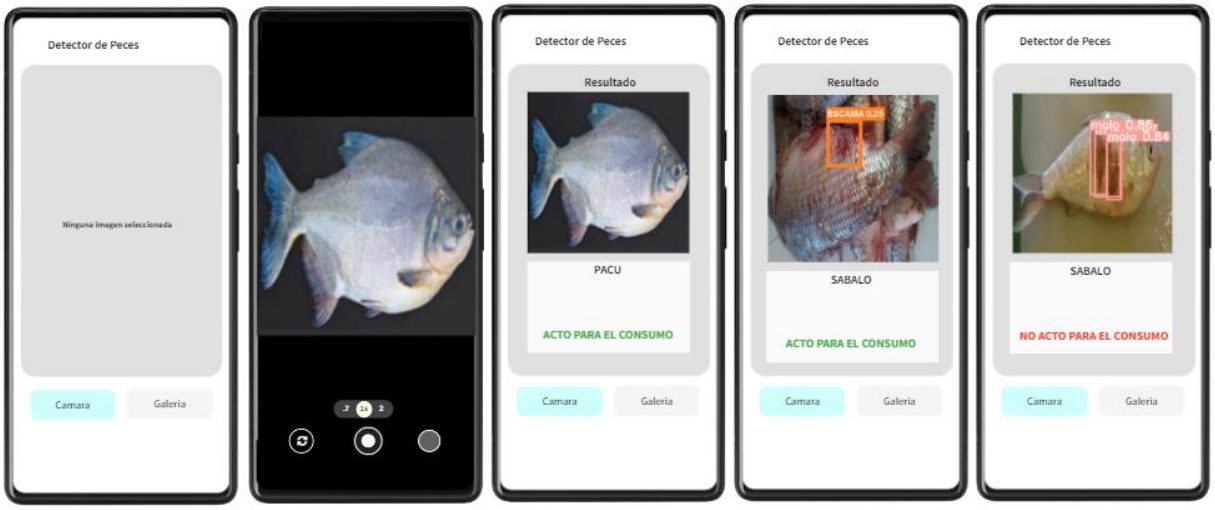
1. Exportar el modelo
2. Prototipo de la aplicación
3. Frameworks y componentes del Software Desarrollados y/o Empleados

1. Exportar el modelo al formato torchscript haciendo el uso de siguiente comando.

```
EXPORTAR A torchscript

!python export.py --weights runs/train/exp/weights/best.pt --include torchscript --img 640 --optimize

export: data=data/coco128.yaml, weights=['runs/train/exp/weights/best.pt'], imgsz=[640], batch_size=1, device=cpu, half=False, inplace=False, train=False, keras=False,
YOLOv5 🚀 v6.1-269-gf76a78e Python-3.7.13 torch-1.11.0+cu113 CPU

Fusing layers...
Model summary: 213 layers, 7015519 parameters, 0 gradients

PyTorch: starting from runs/train/exp/weights/best.pt with output shape (1, 25200, 7) (13.8 MB)

TorchScript: starting export with torch 1.11.0+cu113...
TorchScript: export success, saved as runs/train/exp/weights/best.torchscript (27.0 MB)

Export complete (3.73s)
Results saved to /content/yolov5/yolov5/runs/train/exp/weights
Detect:          python detect.py --weights runs/train/exp/weights/best.torchscript
PyTorch Hub:     model = torch.hub.load('ultralytics/yolov5', 'custom', 'runs/train/exp/weights/best.torchscript')
Validate:        python val.py --weights runs/train/exp/weights/best.torchscript
Visualize:       https://netron.app
```

2. Prototipo de la aplicación Se diseñó el prototipo de la aplicación haciendo el uso de la herramienta MockFlow.

La aplicación es de una sola pantalla, que consta de 2 botones, botón de galería y botón de cámara. Que consiste en seleccionar o capturar una imagen, donde la imagen es enviada al modelo donde la respuesta será la clasificación y la detección de partes en mal estado de los pescados.



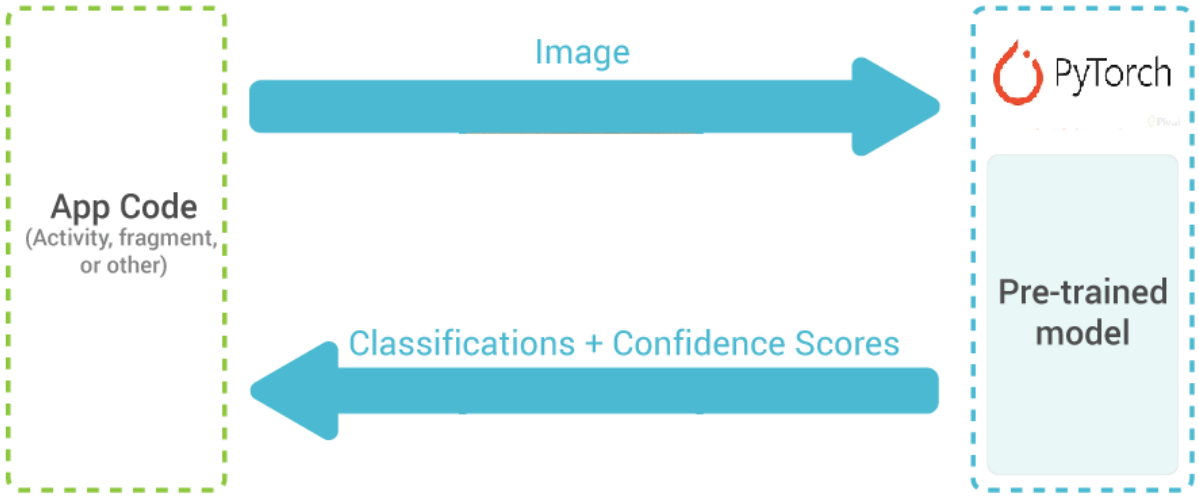
3. Frameworks y componentes del Software Desarrollados y/o Empleados

Frameword de Flutter que es un SDK desarrollado por Google para crear aplicaciones móviles tanto para Android como para IOS (Apple). Juntamente se hiso usó los paquetes de **pytorch_lite y image_picker.**

pytorch_lite: paquete flutter para ayudar a ejecutar la clasificación de modelos pytorch lite y yolov5

image_picker: Un complemento de Flutter para iOS y Android para seleccionar imágenes de la biblioteca de imágenes y tomar nuevas fotos con la cámara.

Patrón arquitectónico empleado



El patrón arquitectónico funciona de la siguiente manera y se representa en la Figura:

1. Se realiza una petición al controlador, por medio de las interfaces de la aplicación.

2. El controlador se comunica con el modelo, quien le retorna al controlador la información solicitada.

3. El controlador le entrega dicha información a la vista, en este proceso, desde el modelo se actualiza la vista con la información solicitada.

8.2 Esquema y descripción de componentes de hardware empleados

8.3 Herramientas utilizadas (Software y Hardware)

Google Colab: Entrenar un sistema se usó Google Colab que es un servicio en la nube en el cual se puede ejecutar notebooks en Python, otros de los motivos por los cuales se eligió este lenguaje de programación, y que te proporciona una GPU con hasta 12GB de memoria de manera totalmente gratuita. Esto te permite desarrollar tu modelo en esta plataforma, la cual tiene una gran cantidad de librerías de Python instalada, y ejecutar tu código desde una GPU permitiéndote entrenar modelos a gran velocidad sin que se desborde la memoria. Los datos necesarios para entrenar el modelo se deben subir a Drive, un almacenamiento en la nube de hasta 15GB que Google te proporciona de manera gratuita. Una alternativa ideal ya que te permite trabajar con librerías de Python muy potentes sin perder tiempo en instalarlas.

Roboflow: es una herramienta para organizar, etiquetar, preparar, versionar y alojar conjuntos de datos para entrenar en YOLOv5.

MockFlow es una herramienta para diseñadores web y desarrolladores que necesiten una herramienta de wireframing en su flujo de trabajo de diseño. Permite crear wireframes en la nube.

Git es un sistema de control de versiones (VCS, por sus siglas en inglés) que te permite guardar tu trabajo, retroceder y avanzar de manera fácil y segura. Git es un VCS distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia.

GitHub El código creado por un desarrollador debe almacenarse, probarse, compilarse, empaquetarse e implementarse para que esté disponible para los clientes; esta es una tarea repetitiva y, al automatizarla, se ahorra mucho tiempo y se reducen los errores humanos.

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Android Studio: La aplicación para dispositivos móviles se desarrolla con Android Software Development Kit.

8.4 Cronograma

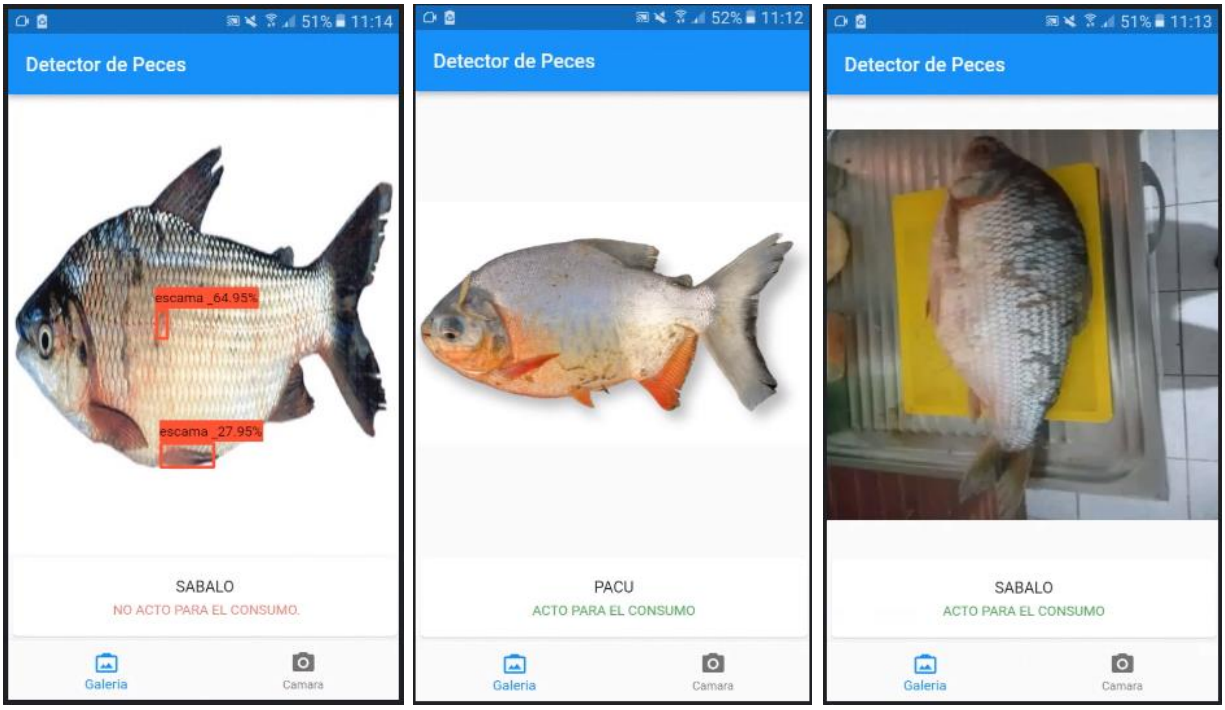
Descripción de la etapa	fecha de inicio	Abril				Mayo				Junio			
Semanas		**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
etapa 1: Construir un data set de imágenes	23/04/2022												
Etapa 2: Entrenar un modelo capaz de detectar un pescado.	07/05/2022												
codificación													
Realizar pruebas													
Corrección de errores													
Etapa 3: Entrenar un modelo capaz de detectar el estado de un pescado.	22/05/2022												
codificación													
Realizar pruebas													
Corrección de errores													
Entrega de los dos algoritmos entrenados													
Etapa 4: Integrar los dos modelos de reconocimiento.	07/06/2022												
Prototipo													
Revisión de fallos													
Integrar algoritmo en un prototipo													
Etapa 5: Desarrollar interfaz móvil.	07/06/2022												
Diseñar													
Codificar													
Integrar a la aplicación													
Realizar pruebas													
Entrega de la interfaz implementada													

8.5 Resultados Obtenidos

Detección de partes en mal estado y la clasificación en imágenes utilizando un modelo entrenado.

Se usaron diferentes tipos de imágenes nuevas para probar el rendimiento del modelo para detectar y clasificar pescados.

La figura muestra que el modelo puede detectar partes en mal estado y clasificar los pescados.



9. CONCLUCIONES

En el estudio actual, YOLOv5 detectó con éxito partes en mal estado y clasificar los pescados. Este es un estudio novedoso haciendo uso de YOLOv5. Se pretende aumentar en mayor medida el número de imágenes y clases para llegar a una mejor y más precisa conclusión.

10. RECOMENDACIONES

Se recomienda aumentar más clases para la clasificación y la detección de los pescados, Todavía hay muchas cosas que se pueden hacer para mejorar el sistema de detección de partes en mal estado y clasificar pescados con YOLOv5.

Para de detección del pescado en mal estado seria el uso de la segmentación semántica, que hasta la fecha de 14/07/2022 ya se liberó YOLOv7. Que tiene un marco de entrenamiento simple y estándar para cualquier tarea de detección y segmentación de instancias, basado en detectron2.

11. CODIGO FUENTE DEL DESARROLLO DE LA APLICACION

Código fuente de la aplicación, modelos ia y el dataset se encuentran en un repositorio público de github: https://github.com/willsonwill/peces-app-yolov5

12. REFERENCIAS BIBLIOGRAFICAS

E. Cengil, A. C. (2021). *Detección de hongos venenosos usando YOLOV5.*

EC., J. (2019). *medlineplus*. Obtenido de https://medlineplus.gov/spanish/ency/article/002851.htm

Kuznetsova A, T. M. (2020). *Detección de manzanas en huertos usando YOLOv3 y YOLOv5 en imágenes generales y de primer plano.*

Roboflow, C. e. (2016). Obtenido de https://colab.research.google.com/drive/1gDZ2xcTOgR39tGGs-EZ6i3RTs16wmzZQ

Thuan, D. (2021). *Evolución del algoritmo YOLO y YOLOv5: el algoritmo de detección de objetos de última generación*. Obtenido de https://www.theseus.fi/handle/10024/452552

Yan. (2021). *Un método de detección de objetivos de Apple en tiempo real para recoger robots basado en YOLOv5 mejorado.*

Yan, P. y. (2020). *Detección de objetos basada en la saturación de la percepción visual multimed. Herramienta. aplicación.*