



Avance 7. Resumen Ejecutivo

Profesores titulares:

Dra. Grettel Barceló Alonso
Dr. Luis Eduardo Falcón Morales

Asesor:

Dr. José Antonio Cantoral Ceballos

Nombre del alumno: Francisco Javier Hernández Camarillo

Matricula: A00998083

Fecha: 10/ Mar/ 2024

Índice

1. Antecedentes	Pág. 3
2. Entendimiento del negocio	Pág. 6
3. Entendimiento de los datos	Pág. 6
4. Exploración de datos	Pág. 7
5. ingeniería de características	Pág. 9
6. Baseline	Pág. 10
7. Explicación de las métricas.	Pág. 13
8. Desarrollo de experimentos	Pág. 17
9. Optimización de hiperparámetros	Pág. 21
10. Modelos Alternativos	Pág. 31
11. Modelo Final	Pág. 33
13. Conclusiones:	Pág. 38
14. Bibliography	Pág. 39
Anexo 1	Pág. 40

1. Antecedentes

Origen:

APTIV se originó como una división de Delphi Automotive en 2017. Delphi Automotive era una empresa bien posicionada en la industria automotriz, conocida por su fabricación de componentes de automóviles y tecnología. La escisión resultó en la formación de dos empresas independientes: Delphi Technologies, enfocada en las soluciones de propulsión, y APTIV, dedicada a la movilidad autónoma y conectada.

Negocio:

APTIV es una empresa líder en tecnología global, especializada en el desarrollo de soluciones para una movilidad más segura, ecológica y conectada. La empresa aborda algunos de los retos más difíciles de la movilidad, incluyendo la creación de vehículos con funciones de seguridad avanzadas, arquitecturas electrificadas y conectividad inteligente.

APTIV se estructura de la siguiente forma:

División "Advanced Safety and User Experience":

Esta división abarca la experiencia de APTIV en software, plataformas de computación centralizadas, sistemas de seguridad avanzados y conducción autónoma. Además, se enfoca en áreas que enriquecen la experiencia dentro del vehículo.

División "Signal and Power Solutions":

APTIV aporta su experiencia como proveedor global de arquitectura y sistemas integradores, ofreciendo sistemas eléctricos de alto voltaje necesarios para la movilidad electrificada y la conectividad de alta velocidad para vehículos altamente automatizados y con muchas funciones.

División "Connected Services":

A través de esta división, APTIV desbloquea valor único mediante robustas capacidades de adquisición de datos y análisis de borde, además de utilizar la nube para entregar conocimientos prácticos.

La posición única de APTIV, como el único proveedor tanto del "cerebro" (software y computación) como del "sistema nervioso" (distribución de energía y datos) de los vehículos, le permite ofrecer soluciones integrales que tienen en cuenta toda la arquitectura del vehículo.

APTIV tiene una presencia global significativa, con más de 200,000 empleados en 131 plantas de fabricación y 11 centros técnicos principales en 48 países (Fig.1). La empresa está comprometida con la sostenibilidad y tiene como objetivo convertirse en una empresa libre de carbono para el año 2040. La amplia gama de negocios y el enfoque en la

innovación continua reflejan la misión de APTIV de impulsar el futuro de la movilidad que es segura, verde y conectada

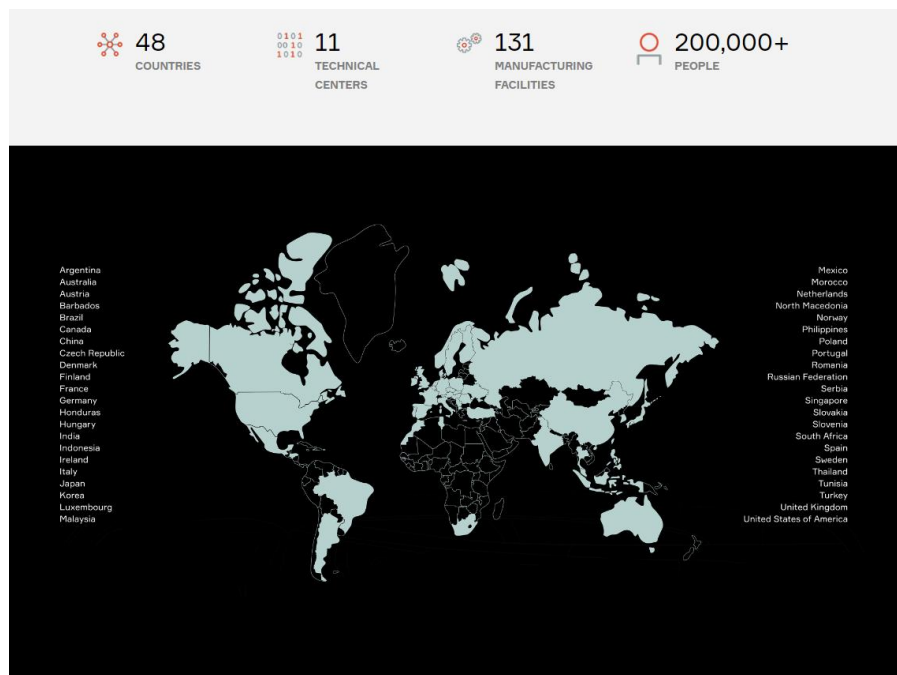


Fig1. Localizaciones de APTIV en el mundo

Proyecto Integrador:

La división de APTIV para la que se desarrollara el proyecto es la de Connection Systems, específicamente para la región de norte América, Connection Systems a su vez es parte de Signal and Power Solutions.

Dentro de connection Systems N.A. se desarrollan diversos conectores (Fig. 2) de bajo voltaje, alto voltaje, especializados y Centrales eléctricas, que satisfacen la demanda de los clientes como son GM, TESLA, FORD, Stellantis y LGM, en el ámbito de electrificación.



Fig. 2.A conectores bajo voltaje



Fig. 2.B conectores Alto voltaje



Fig. 2.D conectores Especializados



Fig. 2.D Centrales eléctricas

Connections systems se encarga de todo el proceso de desarrollo del producto, desde el diseño del producto, manufactura y pruebas de validación, cuando el producto físico cumple con los requisitos del cliente y estándares de la industria automotriz se culmina el proyecto.

El Proyecto integrador **Smart Finite Element Analyzer**, pretende ser un soporte para los ingenieros CAE, los cuales en la fase del diseño realizan validaciones virtuales a los diseños 3D.

Los Ingeniero CAE, usan Software de análisis de elemento finito como ABAQUS, ANSYS y ALTAIR, para analizar los modelos 3D de los conectores (fig. 3), obtienen la geometría, aplican el material adecuado, aplican las cargas a las que se sometería el conector en las pruebas de validación y corren la simulación. Después de obtener los resultados los ingenieros CAE analizan los diferentes resultados como fuerzas, desplazamientos, deformaciones y esfuerzos en los cuales se basan para emitir un reporte en el cual aprueban o rechazan el diseño.

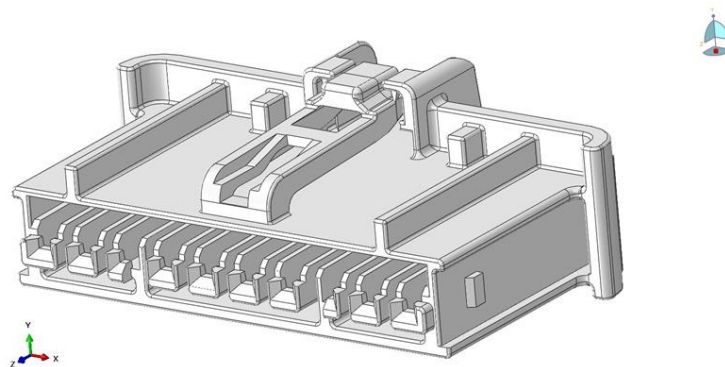


Fig. 3. Conector eléctrico.

Con ayuda del Smart Finite Element Analyzer los ingenieros CAE podrán incrementar su productividad ya que el asistente tiene como objetivo la automatización del proceso de simulación para el Flex lock que sujetan las terminales eléctricas de los conectores. Si bien esto es solo un tipo de componente de la diversa gama de partes que analizan, es un componente común y tener un asistente IA que realice el proceso de simulación para este tipo de componentes, reduce el tiempo de entrega de resultados el ingeniero de producto.

Por lo cual el proyecto integrador Smart finite element analyzer busca impactar la productividad de los Ingenieros CAE que dan servicio a la región de North América de Connection Systems. Dependiendo del éxito de la aplicación esta se puede extender a otras regiones como Europa y Asia-Pacífico que también desarrollan conectores eléctricos.

2. Entendimiento del negocio

¿Qué es lo que se intenta resolver?

El Smart Finite Analyzer busca ser un soporte para el ingeniero CAE, pretende automatizar el proceso de análisis de uno de sus componentes. El ingeniero CAE al tener esta simulación automatizada se enfocaría a analizar los resultados y dar su dictamen de aprobación. Lo que intenta resolver son los tiempos largos de entrega de resultados al ingeniero de producto, minimizando los tiempos de procesamiento de la simulación.

¿Por qué es importante resolver este problema?

Dentro de las simulaciones que se hacen para la cavidad del conector que sujetara la terminal eléctrica es verificar que el Flex lock sea estructuralmente adecuado y cumpla con las cargas de flexión a las que se someterá cuando la terminal pase sobre él y lo deforme.

Es importante resolver este problema ya que la prueba de deflexión es la simulación de entra del Flex lock después se somete a otras pruebas para validarlo completamente. Por lo cual al tener una respuesta más rápida de los resultados del Flex lock ayuda a reducir los tiempos de diseño y además con la automatización del proyecto, reduce los posibles errores humanos a la hora de crear el modelo de simulación.

¿Cuál es la meta prevista?

La meta prevista es que el Smart Finite element analyzer sirva como soporte para el ingeniero CAE y incremente la productividad del personal, lo cual va a mejorar los Kpis del departamento y mejorara los tiempos de desarrollo del diseño.

3. Entendimiento de los datos

- Descripción de los datos: ¿Cuál es el contexto de estos datos? ¿Qué contienen?

Los datos que se usaran para el desarrollo de la aplicación de IA son imágenes y modelos 3D de conectores eléctricos.

- Técnica de ML: Supervisado (Regresión, Clasificación), No-supervisado, Profundo

La técnica de IA propuesta a usar es visión computacional y Aprendizaje supervisado. Visión computacional para el procesamiento de imágenes de los conectores y aprendizaje supervisado para la identificación del área/componente principal a analizar.

- Identificación de las variables: ¿Cuáles son las entradas y la salida?

Las variables de entrada son imágenes y modelos 3d y las variables de salida será las coordenadas donde se identificó la región a analizar.

4. Exploración de datos

Se consiguió una base de datos de +100 modelos 3d a los cuales se manipularon con el software siemens NX para obtener la imagen frontal donde se observa el flex lock.

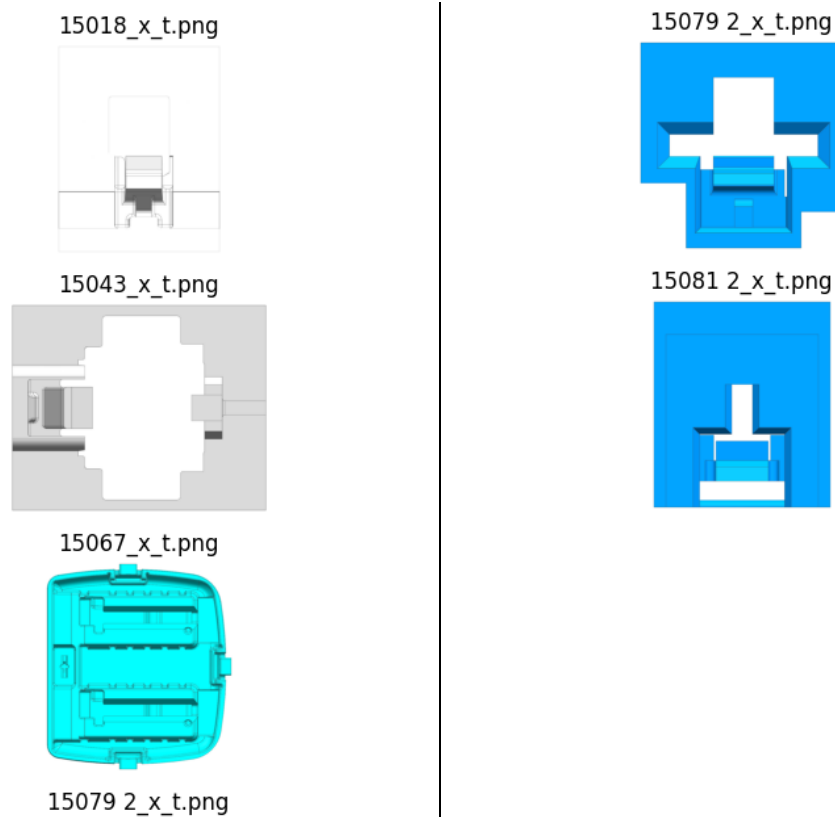


Fig.4 Ejemplo de las imágenes de la base de datos

Se analizaron las imágenes con diferentes técnicas como la detección de bordes de canny para empezar a ver que problemas podrían surgir analizando este tipo elementos de la base de datos al momento de realizar una segmentación de la imagen.

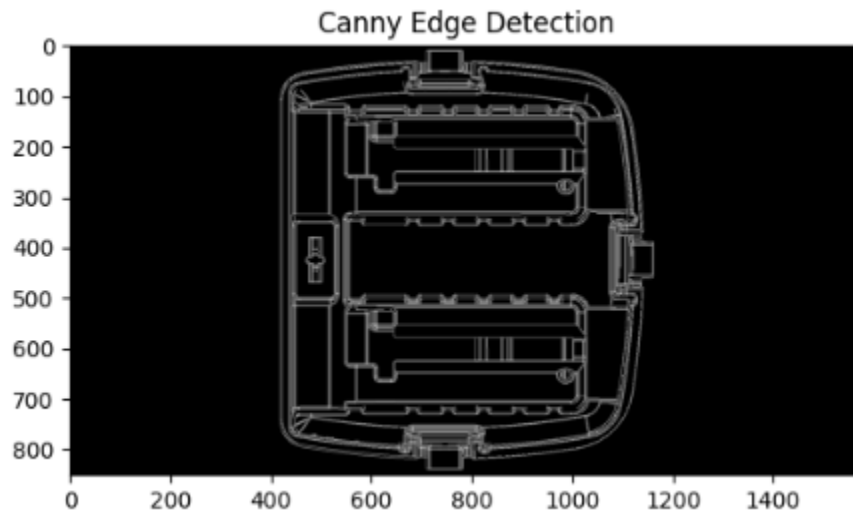


Fig.5 Bordes de detectados

Se analizo el ancho y alto de todas las imágenes para normalizar el tamaño de entrada.

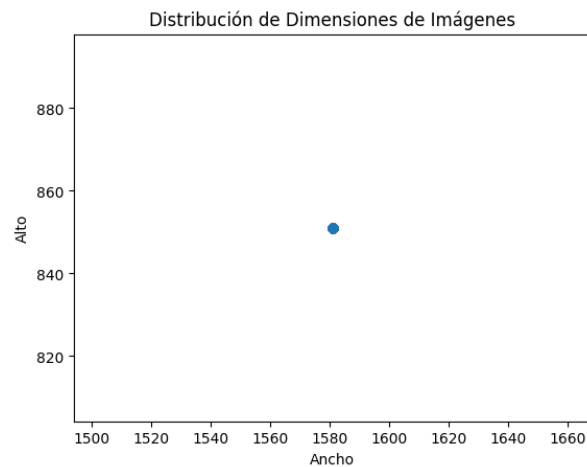


Fig.6 Alto y ancho de las imágenes.

Ya que todos los archivos se obtuvieron de una misma fuente el alto y Ancho fue el mismo para todos. Alto con 851 pixeles y ancho con 1581 pixeles

5. ingeniería de características

5.1 Preparación de datos

El objetivo es crear un conjunto de datos de entrenamiento robusto y bien etiquetado para que el modelo YOLO pueda detectar objetos con precisión.

LabelMe es una aplicación de anotación de imágenes para Python que permite la anotación de imágenes con etiquetas de objetos.

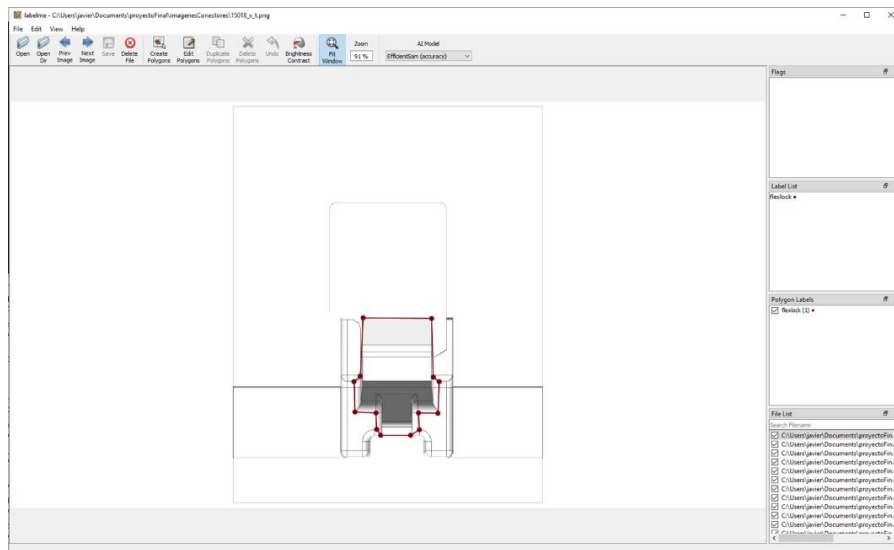


Fig.6 Alto y ancho de las imágenes.

LabelMe no guarda las anotaciones en el formato específico de YOLO por defecto, guardamos las anotaciones en formato .json. Podemos revisar el resultado del etiquetado con el siguiente comando en consola

```
labelme_draw_json C:/Users/javier/Documents/proyectoFinal/imagenesConectores/15117_x_t.json
```

Abajo se muestran algunas capturas que se obtuvieron con este Código.

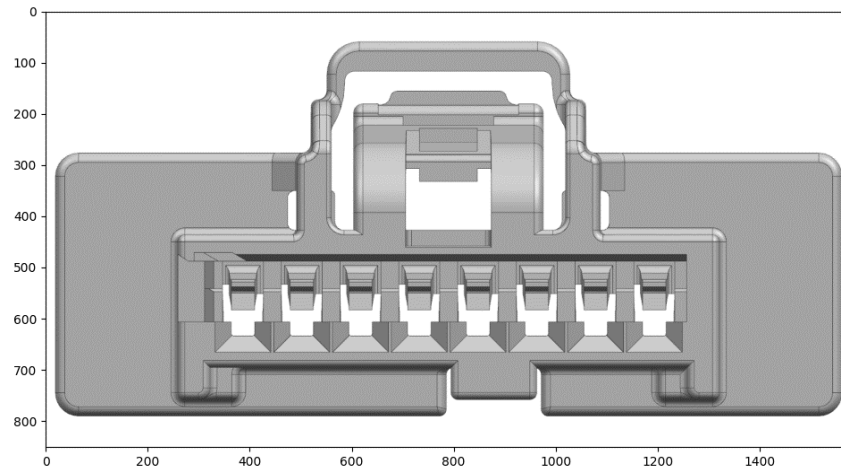


Fig.7 Imagen original

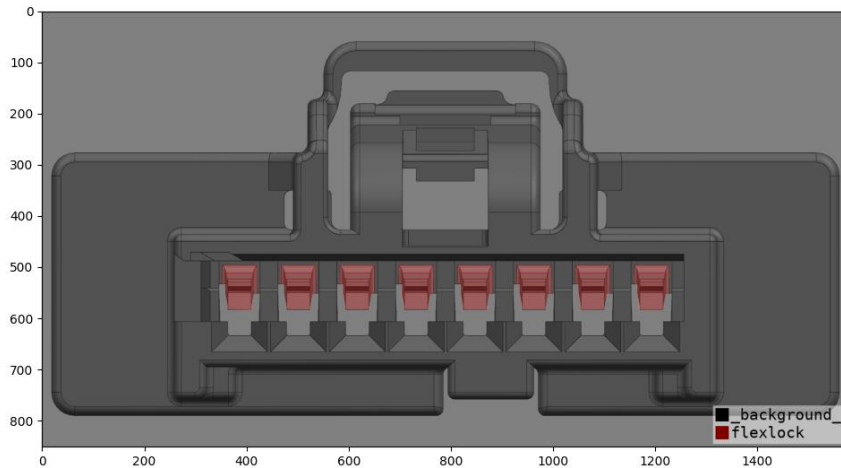


Fig.8 Imagen etiquetada

6. Baseline

Usaremos el framework YOLO para realizar el proceso de aprendizaje del modelo para que pueda identificar los flexlock.

YOLO (You Only Look Once), es un modelo de detección de objetos y segmentación de imágenes, fue desarrollado por Joseph Redmon y Ali Farhadi en la Universidad de Washington. Lanzado en 2015, YOLO ganó popularidad rápidamente por su gran velocidad y precisión.

YOLOv8 es la última versión de YOLO de Ultralytics. Como modelo de vanguardia y de última generación (SOTA), YOLOv8 se basa en el éxito de las versiones anteriores, introduciendo nuevas funciones y mejoras para aumentar el rendimiento, la flexibilidad y

la eficacia. YOLOv8 admite una gama completa de tareas de IA de visión, como la detección, la segmentación, la estimación de la pose, el seguimiento y la clasificación. Esta versatilidad permite a los usuarios aprovechar las capacidades de YOLOv8 en diversas aplicaciones y dominios.

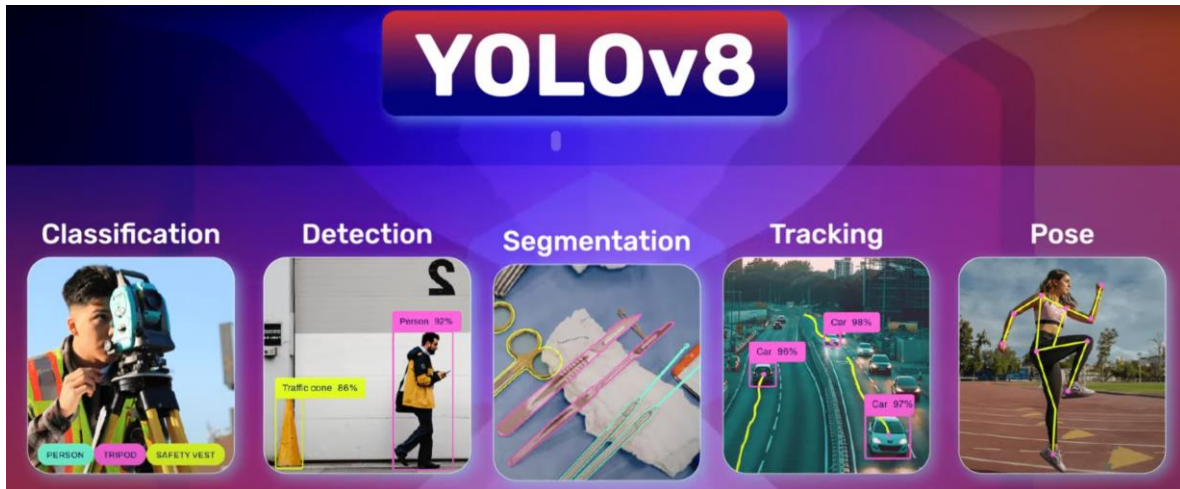


Fig.9 YOLOv8

6.1 Detección con YOLOv8

La detección de objetos es una tarea que consiste en identificar la ubicación y la clase de los objetos en una imagen o secuencia de vídeo.

La salida de un detector de objetos es un conjunto de cuadros delimitadores que encierran los objetos de la imagen, junto con etiquetas de clase y puntuaciones de confianza para cada cuadro. La detección de objetos es una buena opción cuando necesitas identificar objetos de interés en una escena, pero no necesitas saber exactamente dónde está el objeto ni su forma exacta.

6.2. Conociendo el programa

Se creó una primera corrida, usando los datos previamente etiquetados, se procedió a hacer el entrenamiento de YOLO una vez que se crearon las carpetas Train y test con imágenes y archivos txt.

Se dividió la base de datos aproximadamente con una proporción de 80/20, 80% los datos de entrenamiento y 20% para datos de validación

Se usó el siguiente comando para entrenar el modelo.

```
yolo task=segment mode=train epochs=100 data=dataset.yaml model=yolov8m-seg.pt  
imgsz=851 batch=8
```

Esto creó una tarea de entrenamiento la cual corre la cantidad de épocas determinadas y arroja el resultado de las predicciones como se muestra en la fig.11

```

from ultralytics import YOLO

# Load a model
model = YOLO('yolov8m.pt', task="detect") # Load a pretrained model (recommended for training)

# Train the model
results = model.train(data=r"C:\Users\javier\Documents\proyectoFinal\segundaCorrida\dataset.yaml", epochs=100, imgsz=851, batch=4)

```

Starting training for 100 epochs...

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/100	0G	1.85	2.981	1.434	88	864: 100% [02:20<00:00, 35.09s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% [01:02<00:00, 12.46s/it]
	all	77	407	0.366	0.233	0.157 0.0586

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
2/100	0G	1.456	1.837	1.205	78	864: 100% [02:05<00:00, 31.26s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% [01:06<00:00, 13.29s/it]
	all	77	407	0.687	0.479	0.597 0.342

Fig.10 Proceso de entrenamiento

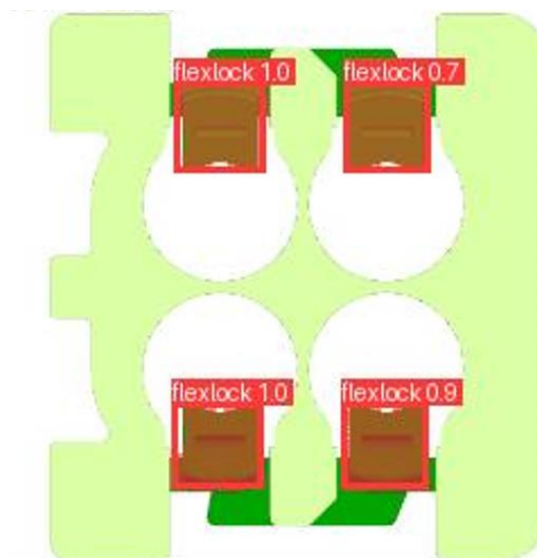


Fig.11 resultado de la primera predicción

7. Explicación de las métricas.

En la imagen fig.11 se puede observar el resultado de la predicción, el valor que se muestra dentro del recuadro rojo es el confidence score el cual se calcula de la siguiente forma:

$$Cs = Pr(object) * IoU$$

7.1 Intersection Over Union (IoU):

IoU se utiliza para evaluar el algoritmo de detección de objetos. Es la superposición entre la verdad fundamental y el cuadro delimitador predicho, es decir, calcula qué tan similar es el cuadro predicho con respecto a la verdad fundamental.

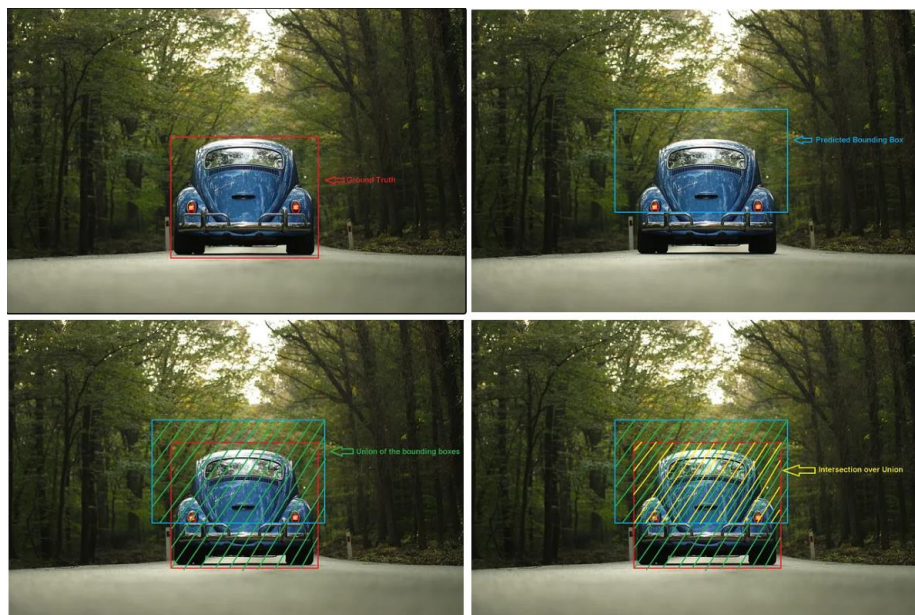


Fig.12 explicación IoU

7.2 Pr(object):

El algoritmo puede encontrar múltiples detecciones del mismo objeto. La Non-max suppression es una técnica mediante la cual el algoritmo detecta el objeto solo una vez. Considera un ejemplo donde el algoritmo detectó tres cuadros delimitadores para el mismo objeto. Los cuadros con sus respectivas probabilidades se muestran en la imagen a continuación.

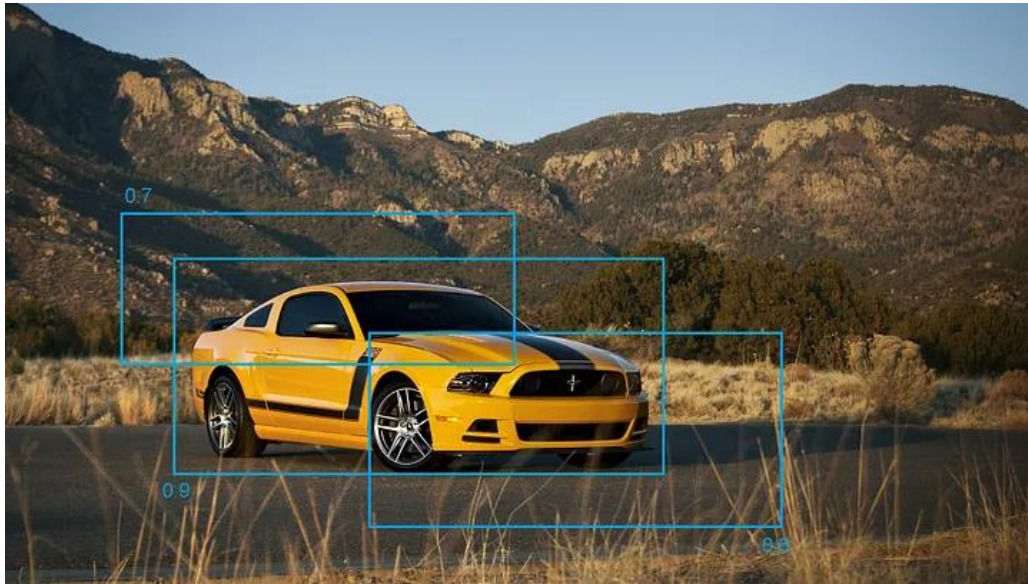


Fig.13 Probabilidades de encontrar un objeto dentro del cuadro

Las probabilidades de los cuadros son 0.7, 0.9 y 0.6, respectivamente. Para eliminar los duplicados, primero vamos a seleccionar el cuadro con la mayor probabilidad y lo presentaremos como una predicción. Luego, eliminaremos cualquier cuadro delimitador con un $\text{IoU} > 0.5$ (o cualquier valor umbral) con la salida predicha. El resultado será:

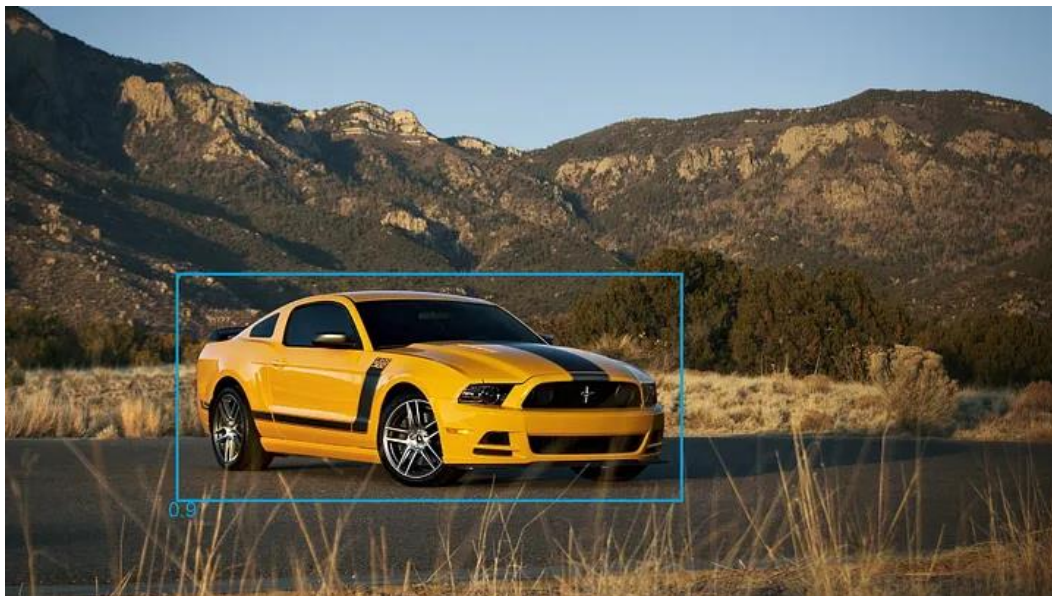


Fig.14 Selección de caja con mejor probabilidad

7.3. Métricas de rendimiento

7.3.1 Precision, Recall and F1

La precisión se establece como una medida fundamental que revela la exactitud de nuestro modelo dentro de una clase específica. Se calculó, como se muestra en la Ecuación (1), como la proporción de Verdaderos Positivos (TP) respecto a la suma de Verdaderos Positivos y Falsos Positivos (FP) para cada nivel del objetivo, es decir, fuego, no-fuego y humo.

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

Formula 1. Precisión.

La precisión indica qué tan confiados podemos estar de que una región detectada, pronosticada para tener el nivel objetivo positivo, realmente tenga el nivel objetivo positivo.

Recall, también conocida como sensibilidad o tasa de verdaderos positivos (TPR, por sus siglas en inglés), indica qué tan confiados podemos estar de que todas las regiones detectadas con el nivel objetivo positivo fueron encontradas. Se definió como la proporción de Verdaderos Positivos (TP) respecto a la suma de Verdaderos Positivos y Falsos Negativos (FN), como se muestra en la Ecuación (2).

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

Formula 2. Recall.

El puntaje F1 es especialmente útil para determinar la confianza óptima que equilibra los valores de precisión y recuperación para ese modelo dado.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Formula 3. F1 score.

7.3.2 Mean Average Precision (mAP)

La Precisión Media Promedio (mAP) es una métrica crucial en la detección de objetos que evalúa el rendimiento del modelo considerando tanto la precisión como la recuperación a través de múltiples clases de objetos. Específicamente, mAP50 se enfoca en un umbral de IoU de 0.5, midiendo qué tan bien un modelo identifica objetos con un solapamiento razonable. Puntuaciones mAP50 más altas indican un rendimiento general superior.

Para proporcionar una evaluación más completa, mAP50-95 extiende la evaluación a una gama de umbrales de IoU de 0.5 a 0.95. Esta métrica es especialmente valiosa para tareas que requieren una localización precisa y detección de objetos de gran detalle.

En la práctica, mAP50 y mAP50-95 ayudan a evaluar el rendimiento del modelo a través de diferentes clases y condiciones, ofreciendo perspectivas sobre la precisión de la detección de objetos mientras se considera el compromiso entre precisión y recuperación. Los modelos con puntuaciones mAP50 y mAP50-95 más altas son más confiables y adecuados para aplicaciones exigentes como la conducción autónoma y la vigilancia de seguridad.

$$\text{mAP}@{\alpha} = \frac{1}{n} \sum_{i=1}^n \text{AP}_i \quad \text{for } n \text{ classes.}$$

Formula 4. mAP50.

7.4 Matriz de confusión

Por lo general, el umbral de conciencia se mantiene $\text{mAP} > 0,5$.

- Verdadero Positivo (True Positive, TP): Se produce cuando se ha detectado el objeto y $\text{IOU} \geq \text{Umbral}$. Es una detección correcta.
- Falso Positivo (False Positive, FP): Se produce cuando el sistema informático detecta el objeto, pero al calcular el IOU se obtiene que $\text{IOU} < \text{Umbral}$. Se trata por tanto de una detección incorrecta.
- Falso negativo (False Negative, FN): Es cuando existe un groundbox en la imagen, pero no se detecta.
- Verdadero Negativo (True Negative, TN): No se utiliza puesto que serían todos los cuadros delimitadores posibles que no contienen y en los que no se detectan objetos. En detección de objetos hay una infinidad de cuadros delimitadores posibles que no se deben detectar en una imagen.

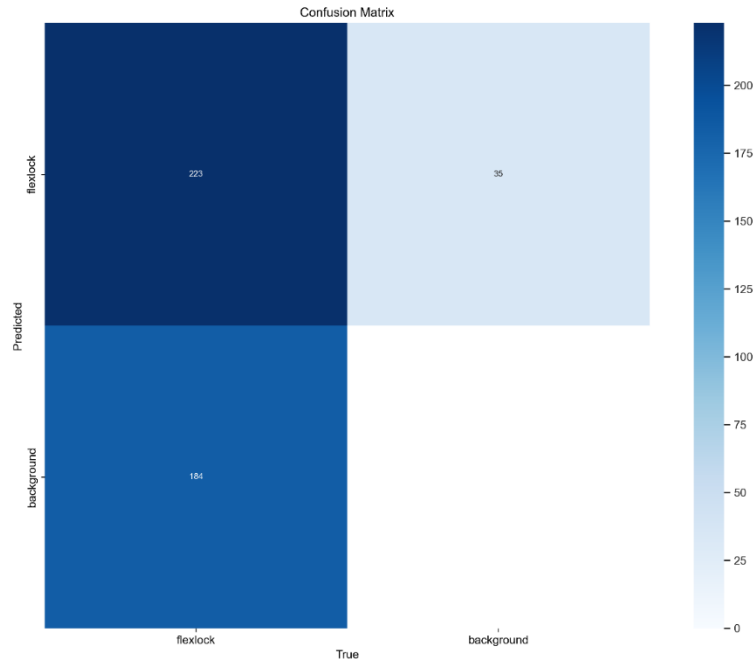


Fig.15 Selección de caja con mejor probabilidad

8. Diseño de experimentos

8.1 Determinando el número de épocas adecuadas.

Una época, "epoch", se refiere a un ciclo completo de entrenamiento de un modelo de aprendizaje automático en un conjunto de datos. Durante una época, el modelo procesa todos los ejemplos del conjunto de datos una vez.

El tamaño de época, o número de épocas, es un parámetro crucial que afecta el rendimiento del modelo de varias maneras:

- **Precisión:** Un mayor número de épocas generalmente conduce a una mayor precisión, ya que el modelo tiene más tiempo para aprender de los datos. Sin embargo, esto también puede llevar a un sobreajuste, donde el modelo memoriza el conjunto de entrenamiento y no se generaliza bien a nuevos datos.
- **Tiempo de entrenamiento:** Un mayor número de épocas también significa un mayor tiempo de entrenamiento, ya que el modelo necesita procesar más datos. Esto puede ser un problema si se tiene un conjunto de datos grande o si se necesita entrenar el modelo rápidamente.
- **Convergencia:** El número de épocas también puede afectar la convergencia del modelo, que es el punto en el que el error del modelo deja de disminuir significativamente. Un número insuficiente de épocas puede impedir que el modelo converja, mientras que un número excesivo de épocas puede llevar a un sobreajuste.

Elegir el tamaño de época adecuado es un proceso de prueba y error.

Se realizó un experimento factorial con un solo factor (tamaño de época) y seis niveles. Se entrenaron seis modelos diferentes, cada uno con un tamaño de época diferente. Las demás variables del modelo de YOLOv8 se mantuvieron constantes.

A continuación, se encuentran los resultados de diferentes métricas y algunos resultados de las predicciones hechas.

Modelo	Épocas	mAP50	Precisión	Recall	T. de entrenamiento
Modelo 1	10	13.7 %	2.19 %	86 %	0.2 hrs
Modelo 2	20	54.8 %.	96.4 %	11.6 %	0.4 hrs
Modelo 3	50	84.3 %	83.1 %	74.7 %	0.9 hrs
Modelo 4	100	85.8 %	78.2 %	81.6 %	1.7 hrs
Modelo 5	200	88.7 %	83.2 %	79.9 %	3.8 hrs
Modelo 6	400	88.8 %	79.3 %	80.3 %	7.5 hrs

Tabla 1. Comparacion del modelo YOLOv8 a diferentes tipos de epocas

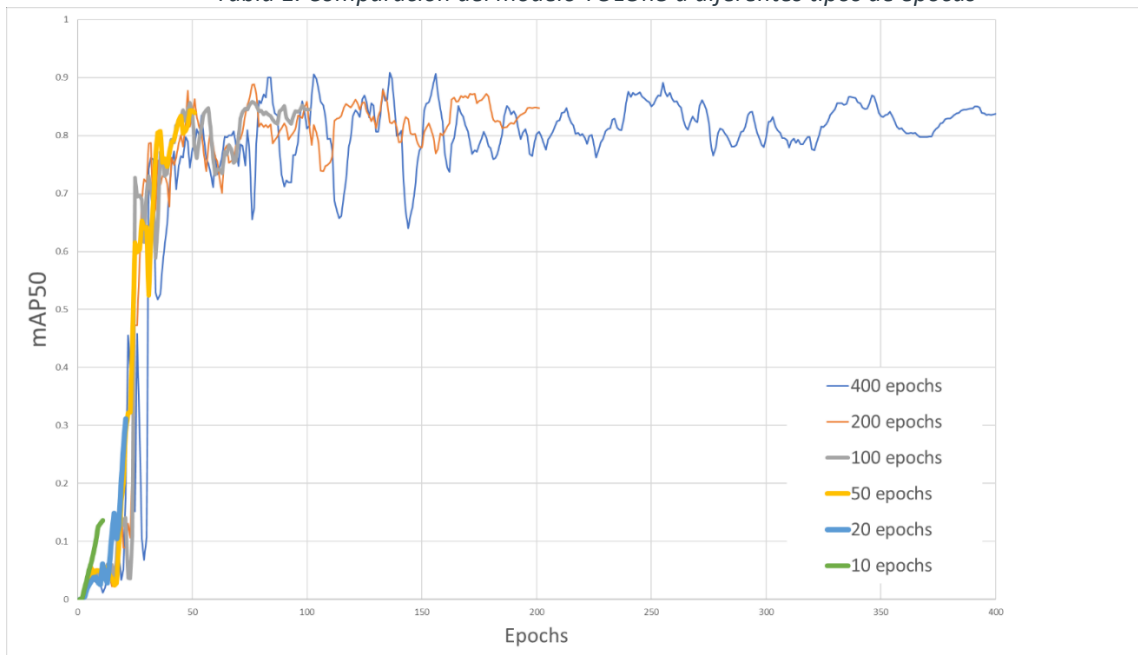


Gráfico 1. Entrenamiento a diferentes valores de épocas.

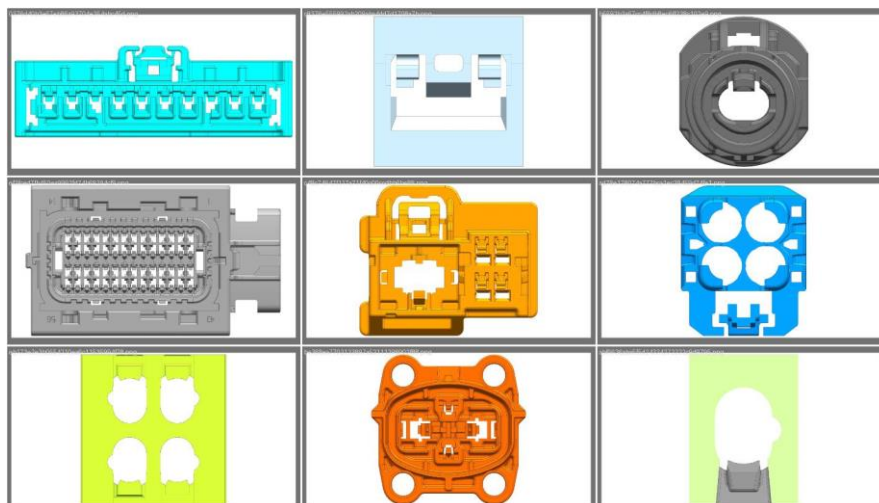


Fig.16 Predicción Modelo de 10 épocas- Detección 0%



Fig.17 Predicción Modelo de 200 épocas- Detección 90%

8.2 Análisis de resultados.

- La precisión y el mAP50 aumentan con el tamaño de época hasta un punto máximo (entre 200 y 400 épocas).
- El recall aumenta con el tamaño de época hasta un punto máximo (alrededor de 100 épocas) y luego comienza a disminuir.
- El tiempo de entrenamiento CPU aumenta proporcionalmente con el tamaño de época

8.3 Verificar sobreentrenamiento

- Se grafican las curvas de error de entrenamiento y validación en función del número de épocas. Un modelo con sobreentrenamiento presentará una curva de error de entrenamiento que disminuye constantemente, mientras que la curva de error de validación comenzará a aumentar después de un cierto número de épocas.

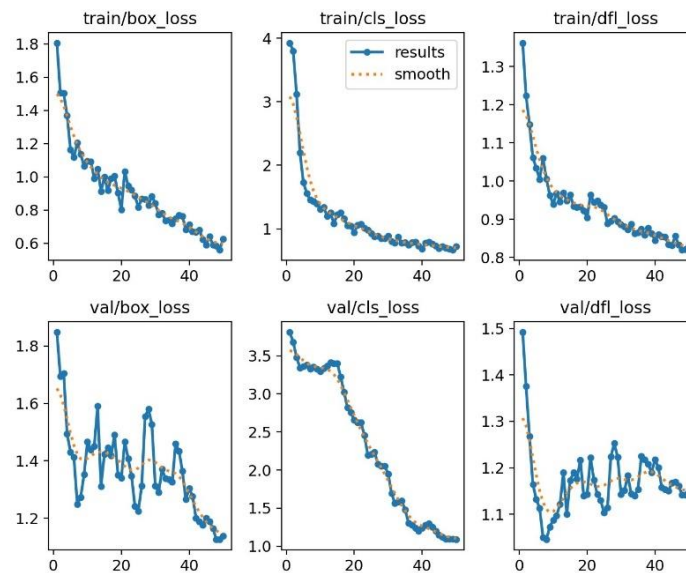


Gráfico 2. Curvas de pérdida. épocas 50

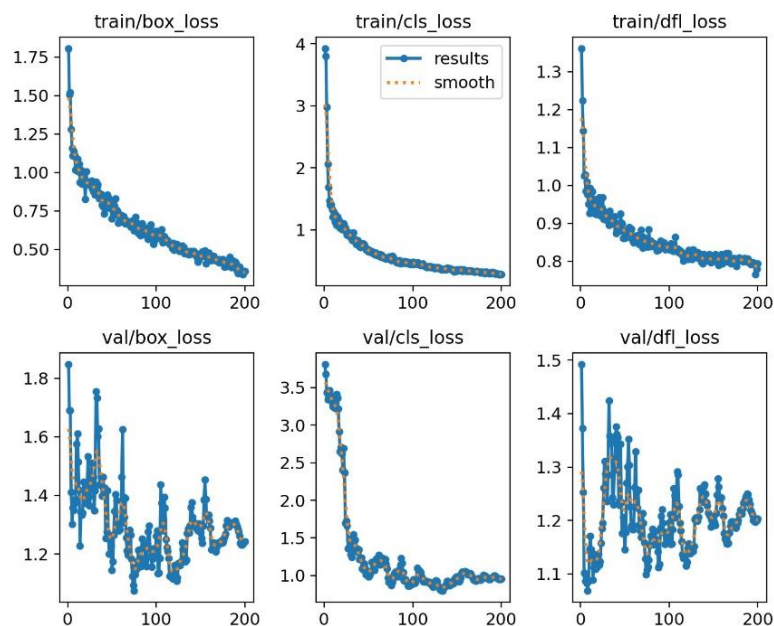


Gráfico 3. Curvas de pérdida. épocas 200

9. Optimización de hiperparámetros

La **optimización de hiperparámetros** es una etapa crucial en el proceso de aprendizaje automático para encontrar la configuración óptima de los hiperparámetros de un modelo. Los hiperparámetros son variables que controlan el comportamiento del modelo durante el entrenamiento, como la tasa de aprendizaje, el tamaño del lote y el número de épocas.

La optimización de hiperparámetros puede mejorar significativamente el rendimiento del modelo, como la precisión, la velocidad de convergencia y la generalización a nuevos datos.

Evita el sobreajuste y el Infra-ajuste. Un ajuste adecuado de los hiperparámetros puede evitar el sobreajuste (el modelo aprende demasiado del conjunto de entrenamiento y no generaliza bien a nuevos datos) y el Infra-ajuste (el modelo no aprende lo suficiente del conjunto de entrenamiento).

9.1 Métodos de Optimización de Hiperparámetros:

1. **Búsqueda manual:**

- Se ajustan manualmente los hiperparámetros y se evalúa el rendimiento del modelo para cada configuración.

2. **Búsqueda en cuadrícula (Grid Search)**

- Evalúa exhaustivamente un conjunto predefinido de combinaciones de hiperparámetros. Crea una cuadrícula con todas las combinaciones posibles a partir de rangos específicos para cada hiperparámetro.

3. **Búsqueda aleatoria (Random Search)**

- Muestra aleatoriamente combinaciones de hiperparámetros de un espacio de búsqueda especificado. Se enfoca en la exploración para encontrar potencialmente buenas combinaciones fuera de una cuadrícula fija.

4. **Búsqueda bayesiana (Bayesian Search)**

- Utiliza un modelo probabilístico (por ejemplo, un proceso gaussiano) para estimar el rendimiento de combinaciones de hiperparámetros no vistas. Selecciona las combinaciones que tienen más probabilidades de mejorar el rendimiento del modelo basado en evaluaciones anteriores.

5. **Búsqueda Yolo (Genetic evolution and mutation)**

- Ultralytics YOLO utiliza algoritmos genéticos para optimizar los hiperparámetros. Los algoritmos genéticos se inspiran en el mecanismo de la selección natural y la genética.
 - **Mutación:** En el contexto de Ultralytics YOLO, la mutación ayuda a buscar localmente el espacio de hiperparámetros aplicando pequeños cambios aleatorios a los hiperparámetros existentes, produciendo nuevos candidatos para la evaluación.
 - **Cruzamiento:** Aunque el cruce es una técnica de algoritmo genético muy popular, actualmente no se utiliza en Ultralytics YOLO para el

ajuste de hiperparámetros. La atención se centra en la mutación para generar nuevos conjuntos de hiperparámetros.

9.2 Ventajas y desventajas.

Metodo	Ventaja	Desventaja
Busqueda manual	<ul style="list-style-type: none"> Tienes control total sobre el proceso de búsqueda y puedes ajustar los hiperparámetros de forma precisa. No requiere librerías adicionales 	<ul style="list-style-type: none"> Puede ser un proceso tedioso y lento, especialmente para problemas con un gran número de hiperparámetros. La elección de los hiperparámetros depende de la experiencia y el juicio del investigador, lo que puede llevar a resultados subóptimos.
Grid Search	<ul style="list-style-type: none"> Garantiza encontrar la combinación óptima dentro de la cuadrícula definida. Útil para comprender las interacciones entre hiperparámetros. 	<ul style="list-style-type: none"> Puede ser computacionalmente costoso, especialmente con cuadrículas de alta dimensión o conjuntos de datos grandes. No explora más allá de la cuadrícula definida, si la combinación óptima se encuentra fuera de ella.
Random Search	<ul style="list-style-type: none"> Más eficiente que la búsqueda en cuadrícula, especialmente con grandes conjuntos de datos o espacios de búsqueda de alta dimensión. Menos propenso a quedarse atascado en óptimos locales. 	<ul style="list-style-type: none"> No hay garantía de encontrar la combinación óptima, pero a menudo puede proporcionar buenos resultados con menos evaluaciones.
Bayesian Search	<ul style="list-style-type: none"> Explora eficientemente regiones prometedoras del espacio de búsqueda. Menos probable que se atasque en óptimos locales en comparación con la búsqueda en cuadrícula. 	<ul style="list-style-type: none"> Puede ser más complejo de implementar que la búsqueda en cuadrícula o aleatoria. Puede requerir más conocimiento del dominio para definir el espacio de búsqueda de manera efectiva.
Genetic evolution	<ul style="list-style-type: none"> La Evolución genética y mutación (EGM) es un método eficiente que explora el espacio de búsqueda de hiperparámetros de forma rápida y efectiva. La implementación de EGM en YOLOv8 es sencilla y no requiere de conocimientos avanzados en optimización. 	<ul style="list-style-type: none"> El rendimiento de la EGM depende de la configuración de los parámetros del algoritmo, como la tasa de mutación y la selección de padres.

Tabla 2. Ventajas y desventajas de las técnicas de optimización

9.1 Optimización de hiperparámetros opción 1.

9.1.1 Evolución genética y mutación

Para este tipo de optimización es necesario especificar las constantes que va a tener durante el proceso, elegimos el número de épocas que se había determinado anteriormente, las iteraciones que tendrá la optimización y el optimizador que recomienda Yolo.

9.1.2 Code:

```
model = YOLO('yolov8n.pt')
model.tune(data='data.yaml', epochs=200, iterations=100, optimizer='AdamW')
```

9.1.3 Resultados

9.1.3.1 Fitness vs iteration

Se trata de un gráfico que muestra la aptitud frente al número de iteraciones. Te ayuda a visualizar el rendimiento del algoritmo genético a lo largo del tiempo.

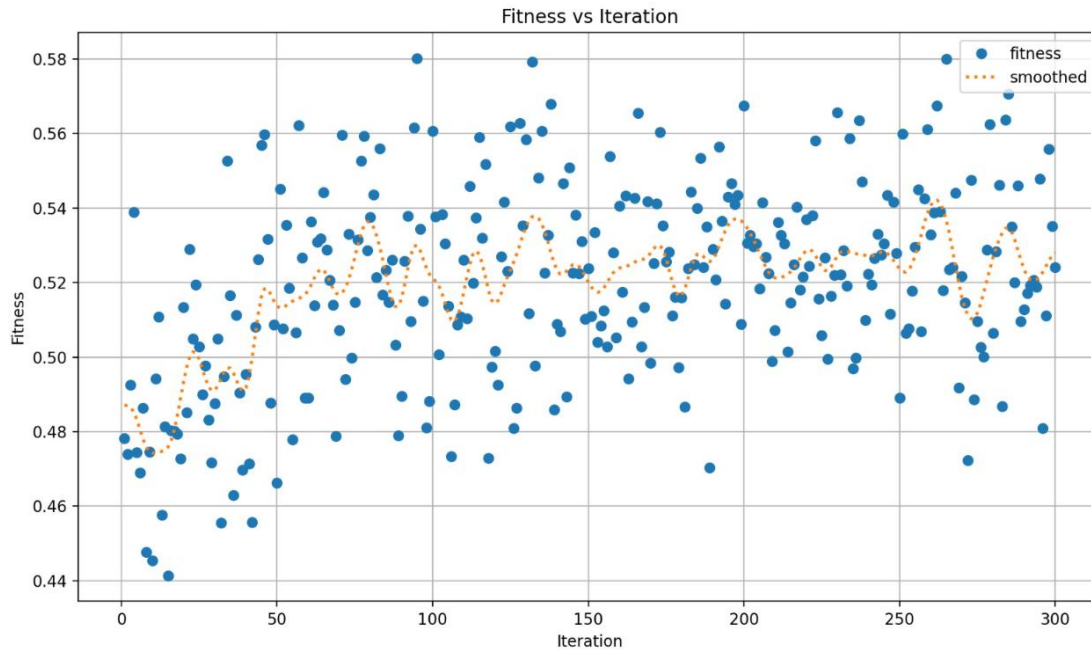


Gráfico 4. Valores fitness

9.1.3.2 Gráfico de dispersión:

Este archivo contiene gráficos de dispersión ayudándote a visualizar las relaciones entre los distintos hiperparámetros y las métricas de rendimiento.

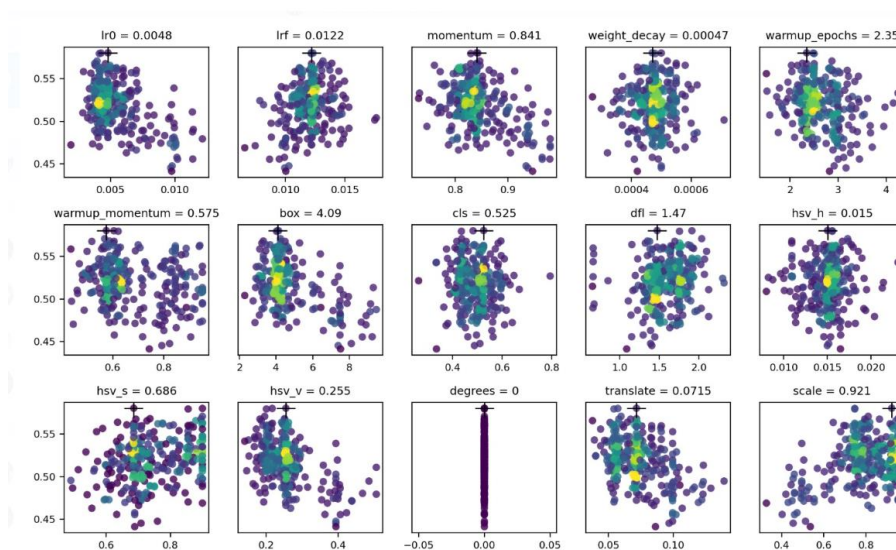


Gráfico 5. Valores de hiperparametros

9.1.3.3 Resultado matriz de confusión

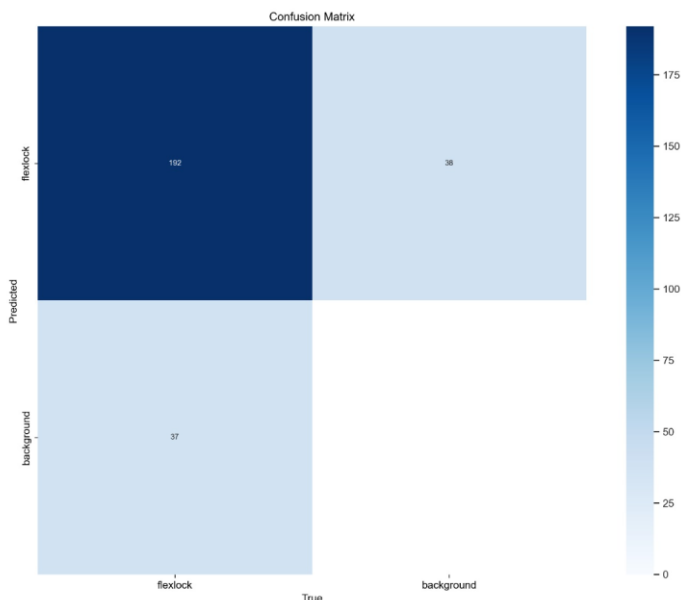


Fig.18 Selección de caja con mejor probabilidad

9.1.3.4 Variables de Salida.

Variable	Best fitness hyperparameters	Variable	Best fitness hyperparameters
lr0	0.00578	hsv_s	0.4575
lrf	0.00893	hsv_v	0.35714
momentum	0.83599	Degrees	0.0
weight_decay	0.00039	translate	0.15217
warmup_epochs	2.72066	scale	0.32105
warmup_momentum	0.95	shear	0.0
box	1.53967	perspective	0.0
cls	0.49185	Flipud	0.0
dfl	1.42627	fliplr	0.14602
hsv_h	0.02335	mosaic	0.92146

Tabla 3. Resultados finales de la hiperparametrización

9.1.3.5 Métricas de la evaluación del modelo con hiperparametros optimizados.

Model	precision	recall	mAP50
geneticMutation.pt	87.8%	79.0%	88.5%

Tabla 4. Métricas de performance

9.1.3.6 Resultados predicción de la validación.

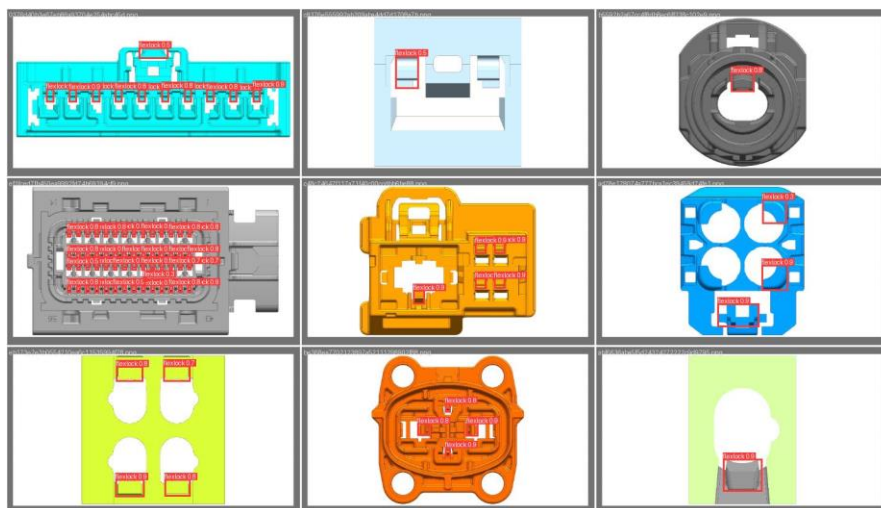


Fig.18 Predicción Modelo de 200 épocas- Detección 90%

9.2 Optimización de hiperparámetros opción 2.

9.2.1 Optimización de hiperparámetros con weights and biases

Weights and biases es una librería de Python que permite hacer una optimización bayesiana además de que guarda cada uno de los resultados para después compararlos

Yolov8 tiene +45 hiperparametros que pueden ser ajustados durante el entrenamiento. Se escogieron los más relevantes para ser ajustados durante la optimización.

- **Épocas (epochs):** Número total de épocas de entrenamiento. Cada época representa una pasada completa por todo el conjunto de datos. Ajustar este valor puede afectar a la duración del entrenamiento y al rendimiento del modelo.
- **Tamaño del lote (batch):** Determina la cantidad de muestras procesadas antes de que el modelo actualice sus pesos.
- **Optimizer:** Activa la salida detallada durante el entrenamiento, proporcionando registros detallados y actualizaciones del progreso. Útil para depurar y supervisar de cerca el proceso de entrenamiento.
- **Tasa de aprendizaje inicial (lr0):** El ajuste de este valor es crucial para el proceso de optimización, ya que influye en la rapidez con que se actualizan las ponderaciones del modelo. Controla cuánto ajustar el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo.
- **Amp:** Permite el entrenamiento Automático de Precisión Mixta (AMP), reduciendo el uso de memoria y posiblemente acelerando el entrenamiento con un impacto mínimo en la precisión.
- **cos_lr:** Utiliza un programador de la tasa de aprendizaje coseno, que ajusta la tasa de aprendizaje siguiendo una curva coseno a lo largo de las épocas. Ayuda a gestionar la tasa de aprendizaje para una mejor convergencia.

- **Tasa de aprendizaje final (lrf):** Tasa de aprendizaje final como fracción de la tasa inicial = $(lr0 * lrf)$, que se utiliza junto con los programadores para ajustar el ritmo de aprendizaje a lo largo del tiempo.
- **Aumento (augment):** Indica si introducir cambios aleatorios en los datos de entrada, aumentando la robustez del modelo.
- **Abandono (dropout):** Es una técnica de regularización para prevenir el sobreajuste.
- **Factor de impulso (Momentum):** Factor de impulso para SGD o beta1 para optimizadores Adam, que influye en la incorporación de gradientes pasados en la actualización actual.
- **Término de regularización (weight_decay):** penaliza los pesos grandes para evitar el sobreajuste.

9.2.2 Valores seteados para la optimización de los hiperparametros.

nombre	valor	nombre	valor
epochs	400	lr0	min': 0.0005, max': 0.005
batch	[10, 15,20]	lrf	min': 0.001, max': 0.01
optimizer	['SGD', 'Adam']	momentum	min': 0.001, max': 0.01
amp	[True, False]	weight_decay	min': 0.001, max': 0.01
cos_lr	[True, False]		

Tabla 5. Grid de parámetros

9.2.3 Objetivo

Nos centramos en maximizar el valor de mAP50, que es métrica recomendada para aplicaciones Yolo de detección.

mAP50 significa Precisión Media Promedio con un IoU de 0.5. En términos simples, evalúa cuán bien las cajas delimitadoras predichas por nuestro modelo se superponen con las cajas delimitadoras reales. Maximizar mAP50 significa que nuestro modelo no solo está detectando objetos, sino que también está ubicándolos con precisión.

Al entrenar nuestro modelo por primera vez con hiperparámetros predeterminados, observé que mAP50 alcanzó su valor óptimo alrededor de la marca de 200 épocas, mientras que mAP50-95 toma más tiempo, incluso con valores mas bajos. Notablemente, un alto mAP50 a menudo indicaba un mAP50-95 igualmente alto. Dado este comportamiento, para fines de optimización de hiperparámetros, elegí centrarme en

maximizar mAP50. Este enfoque nos permite acortar cada evaluación, permitiendo una exploración más eficiente de diversas combinaciones de parámetros.

9.2.4 Resultados optimización bayesiana

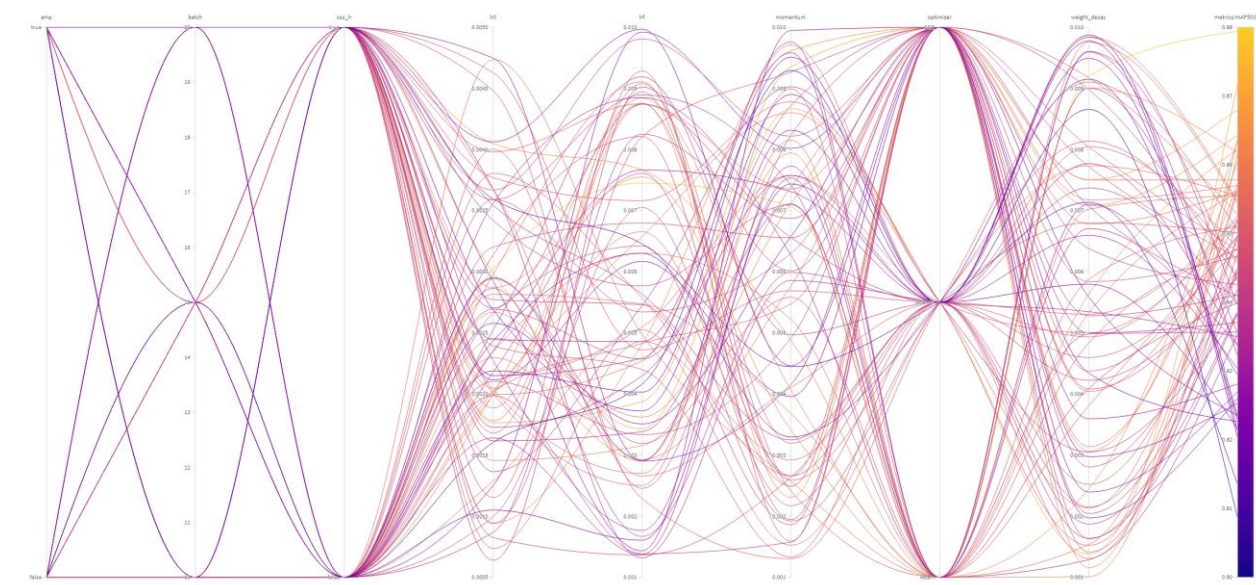


Fig. 19. Iteraciones de la optimización bayesiana

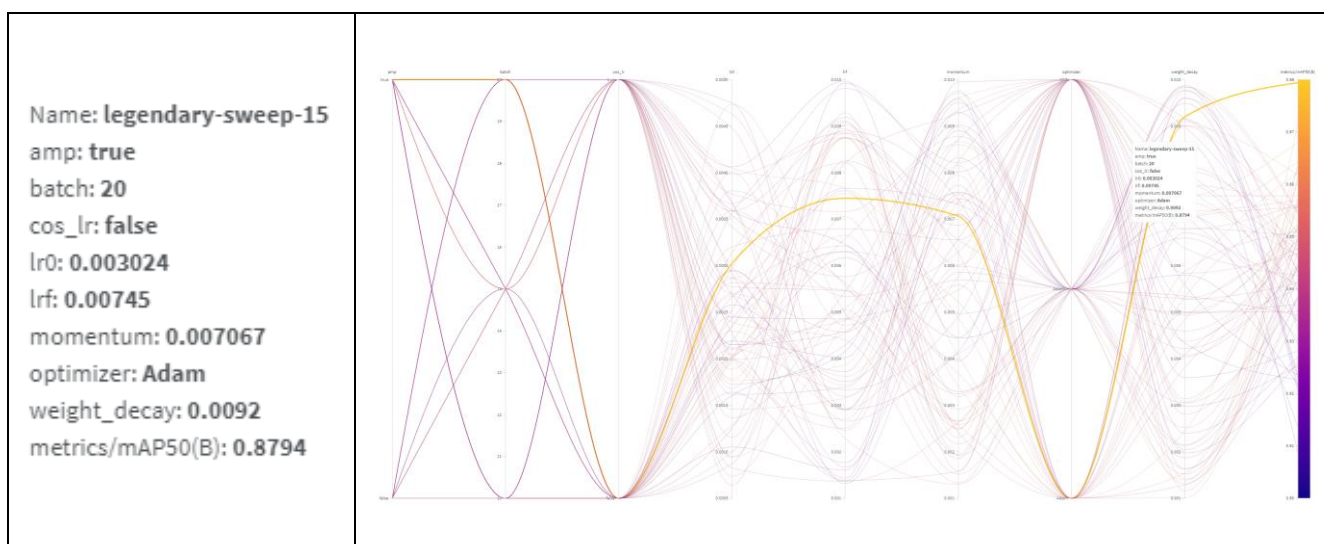


Fig. 20. Mejor iteración mAP 87.9 %

9.2.5 Importancia de parámetros:

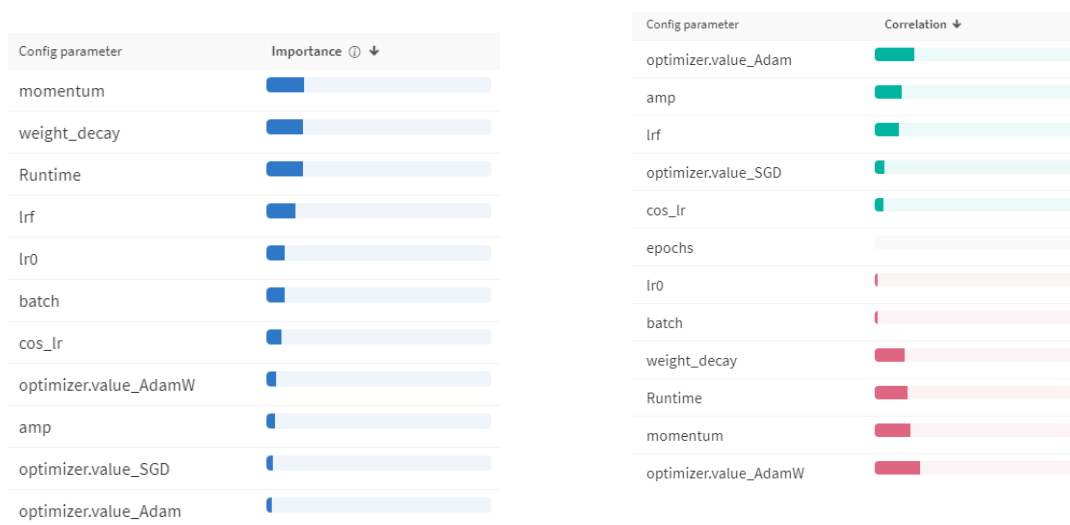


Fig. 21. Importancia de hiperparámetros y correlación

9.2.6 Hiperparámetros finales:

nombre	valor	nombre	valor
epochs	400	lr0	0.003
batch	20	lrf	0.007
optimizer	Adam	momentum	0.007
amp	True	weight_decay	0.009
cos_lr	False	mAP50	87.9%

Tabla 6. Parámetros de salida

9.2.7 Métricas de la evaluación del modelo con hiperparámetros optimizados.

Model	precision	recall	mAP50
Beyesian.pt	85.4%	69.0%	82.4%

Tabla 7. Parámetros de salida

9.2.8 Resultados predicción de la validación.

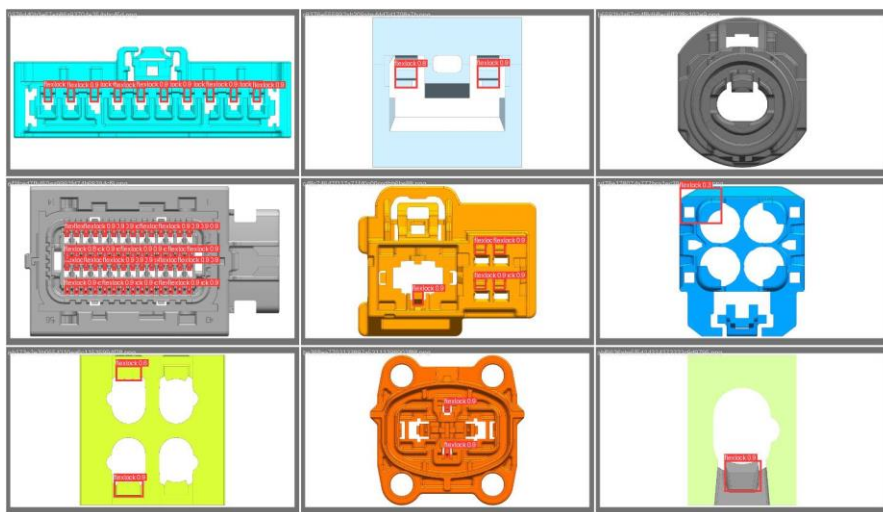


Fig. 21. Predicción

La optimización de hiperparámetros puede mejorar significativamente el rendimiento de tu modelo. Con herramientas como W&B y métodos como la optimización bayesiana, este proceso se vuelve sistemático y revelador. Mientras que algunos parámetros como el optimizador y la tasa de aprendizaje son cruciales, otros tienen un impacto menor. Esto enfatiza la necesidad de una sintonización específica para la tarea en lugar de las mejores prácticas generales de aprendizaje profundo. Al desplegar un modelo, es esencial ponderar las compensaciones entre el rendimiento máximo de validación y la robustez en variadas situaciones del mundo real. En algunos casos, un modelo con una precisión de validación ligeramente menor pero mejor generalización debido a la regularización podría ser la mejor opción.

10. Modelos Alternativos

10.1 Entrenamiento

Dentro de los modelos de detección YOLO existen varias alternativas que se describen a continuación:

1. **yolov8n.pt:** Es el modelo más pequeño y rápido, ideal para dispositivos móviles o aplicaciones donde la velocidad es crucial. Sin embargo, su precisión es menor en comparación con los modelos más grandes.
2. **yolov8s.pt:** Ofrece un buen equilibrio entre velocidad y precisión, siendo adecuado para una amplia gama de aplicaciones.
3. **yolov8m.pt:** Brinda mayor precisión que el yolov8s.pt, aunque con un ligero sacrificio en la velocidad. Es una buena opción para tareas donde se requiere un balance entre rendimiento y exactitud.

4. **yolov8l.pt**: Se destaca por su alta precisión, pero con una velocidad considerablemente menor. Se recomienda para tareas donde la precisión es crítica, como la inspección industrial o el análisis médico.
5. **yolov8x.pt**: Es el modelo más grande y preciso de la familia YOLOv8, pero también el más lento. Se utiliza en casos donde se necesita la máxima precisión posible, como en la conducción autónoma o el reconocimiento facial.

10.2 Resultados de los entrenamientos

Las condiciones de entrenamiento fueron iguales para todos los modelos alternativos.

Abajo en la tabla muestra el rendimiento que tuvo cada modelo

<i>Model</i>	<i>precision</i>	<i>recall</i>	<i>mAP50</i>	<i>Tiempo Eje.</i>
<i>yolov8n.pt</i>	79.3 %	80.3 %	88.8 %	7.5 hrs
<i>yolov8s.pt</i>	78.2 %	74.7 %	84.3 %	16.0 hrs
<i>yolov8m.pt</i>	85.2%	75.7%	85.9 %	24.5 hrs
<i>yolov8l.pt</i>	82.6 %	78.8 %	84.6 %	32.0 hrs

Tabla 8. Resultados de los modelos alternativos

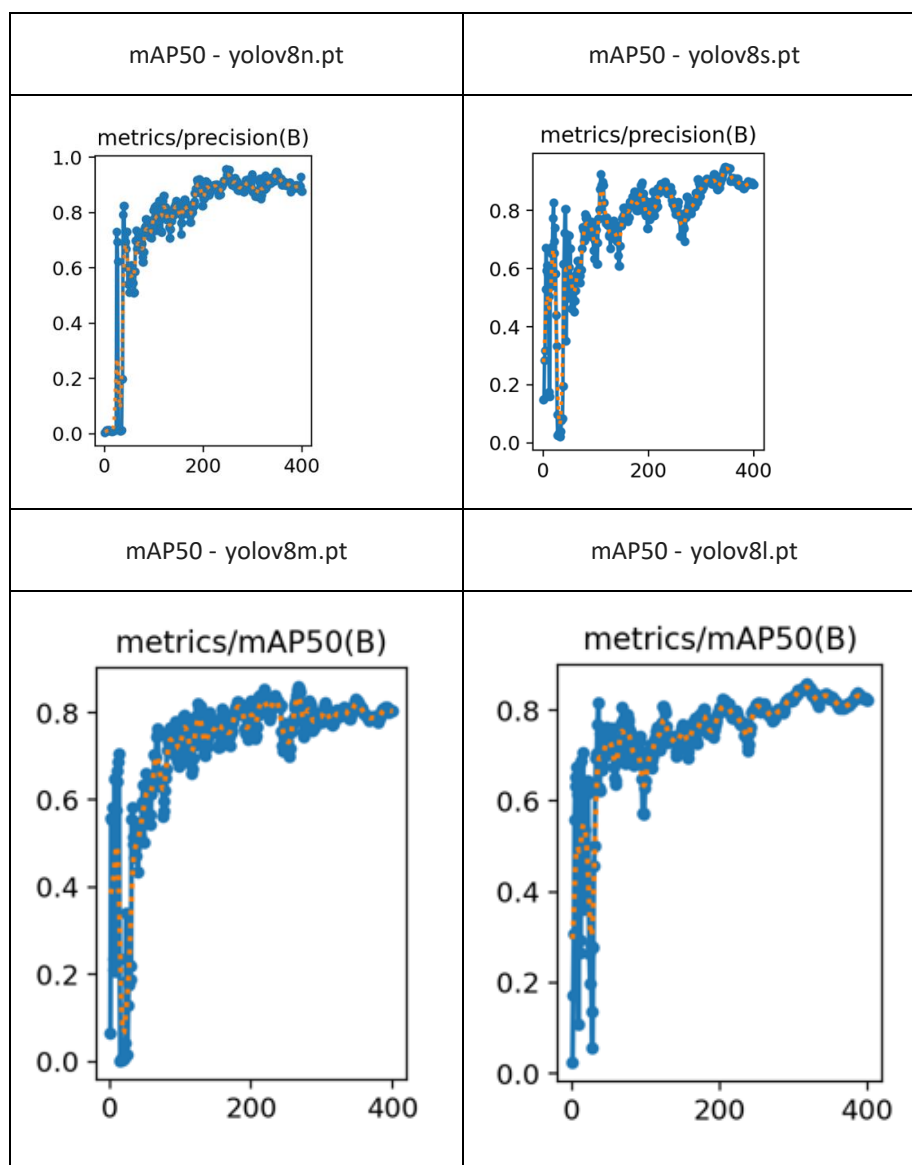


Gráfico 6. Métricas de precisión para los cuatro modelos alternativos

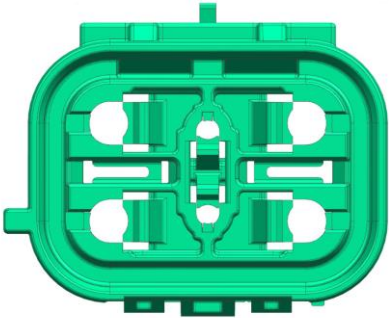
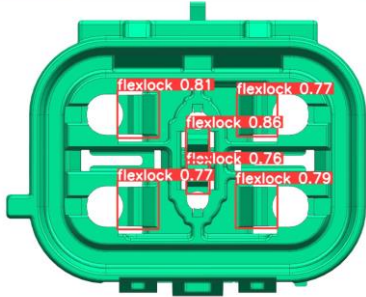
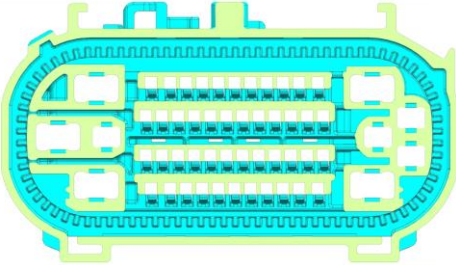
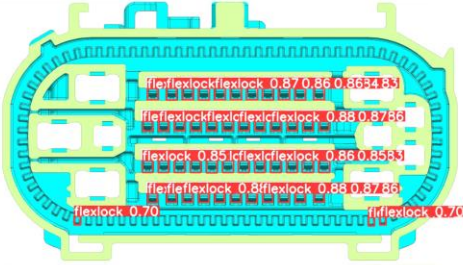
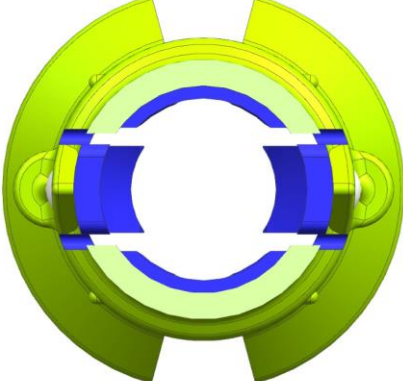
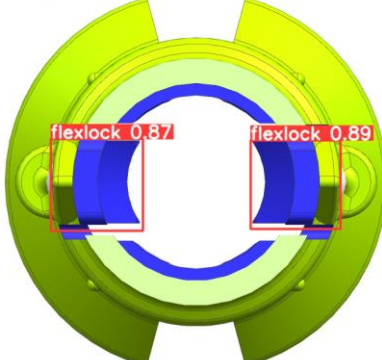
Debido a que los métricos de mAP50 resultaron similares a al modelo inicial usado, optamos por este modelo para este proyecto ya que los tiempos de entrenamiento resultaron adecuados.

11. Modelo Final

Selecione como mejor modelo al archivo best.pt que fue el resultado de la optimización por mutación genética de Yolo.

Procedemos a evaluarlo con imágenes que no se habían metido antes en el modelo de entrenamiento.

11.1 Predicciones

Imagen	predicción
	
	
	

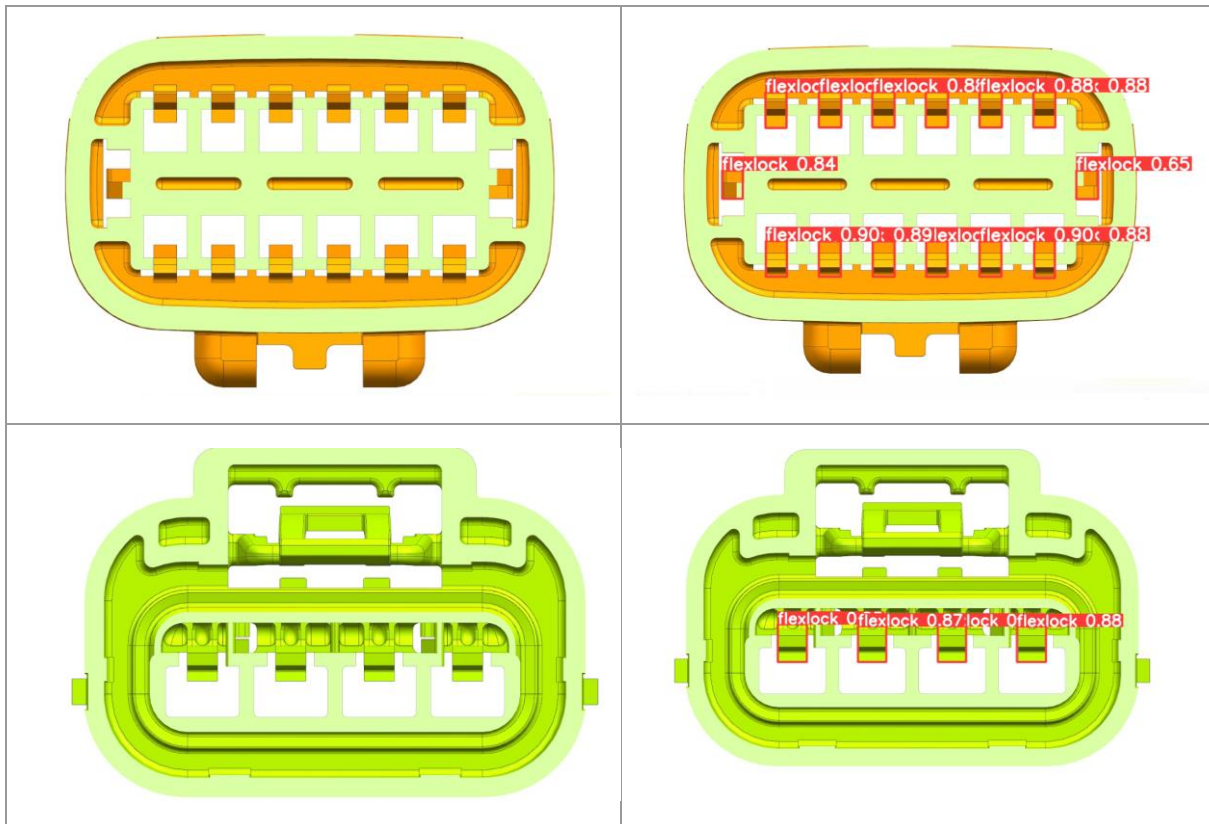


Fig. 22. Predicción de modelos

12. Implementación

En la fase de implementación de este proyecto, nos centramos en el desarrollo y la integración de una aplicación diseñada específicamente para optimizar y automatizar el proceso de simplificación de geometría dentro del software de simulación Abaqus. Abaqus, es un software de elemento finito que permite la integración de código Python para el desarrollo de aplicaciones como esta.

La aplicación desarrollada actúa como un complemento que se integra de manera fluida con el entorno de Abaqus, permitiendo a los usuarios realizar la simplificación de la geometría de una forma mucho más eficiente. Este complemento utilizará el mejor modelo de IA artificial creado y descrito anteriormente.

12.1 Predicción.

Empezamos por realizar una predicción con el modelo ya entrenado de Yolo de la siguiente forma

```
# Run inference on 'bus.jpg' with arguments
results = model.predict(image_path, save=True, imgsz=320, conf=0.5)

boxes = results[0].boxes

for box in boxes:
    print(box.xyxy)
```

Esto nos arroja el siguiente resultado

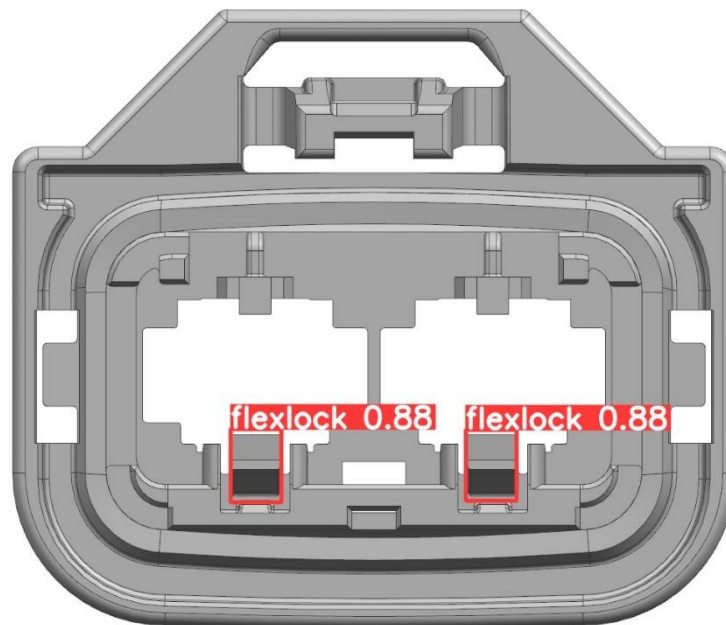


Fig. 23. Confidence score de la predicción

Con los valores de los tensores de cada caja de predicción en la imagen.

Resultado 1: tensor ([913.8806, 573.7177, 981.5040, 668.0735])

Resultado 2: tensor ([632.5806, 573.7177, 721.5040, 668.0735])

Cada tensor contiene dos puntos (x,y) puntos que coinciden con las coordenadas de las esquinas superior izquierda y la inferior derecha del cuadrado resultante.

12.2 Transformación de coordenadas

Una vez obtenida el cuadrado de predicción se procede a transformar los valores de los píxeles del cuadrado en coordenadas del sistema donde se encuentra el modelo 3D ya que las obtenidas del modelo YOLO se encuentran en píxeles

	Punto 1, Caja 1 (esquina superior izquierda)	Punto 2, Caja 2 (esquina inferior derecha)
<i>Píxeles de imagen X, Y</i>	913.88, 573.71	981.5040, 668.0735
<i>Coordenadas X, Y, Z Abaqus</i>	8.64, -2.52,0	6.28, -5.40,0

Tabla 9. Transformación de coordenadas

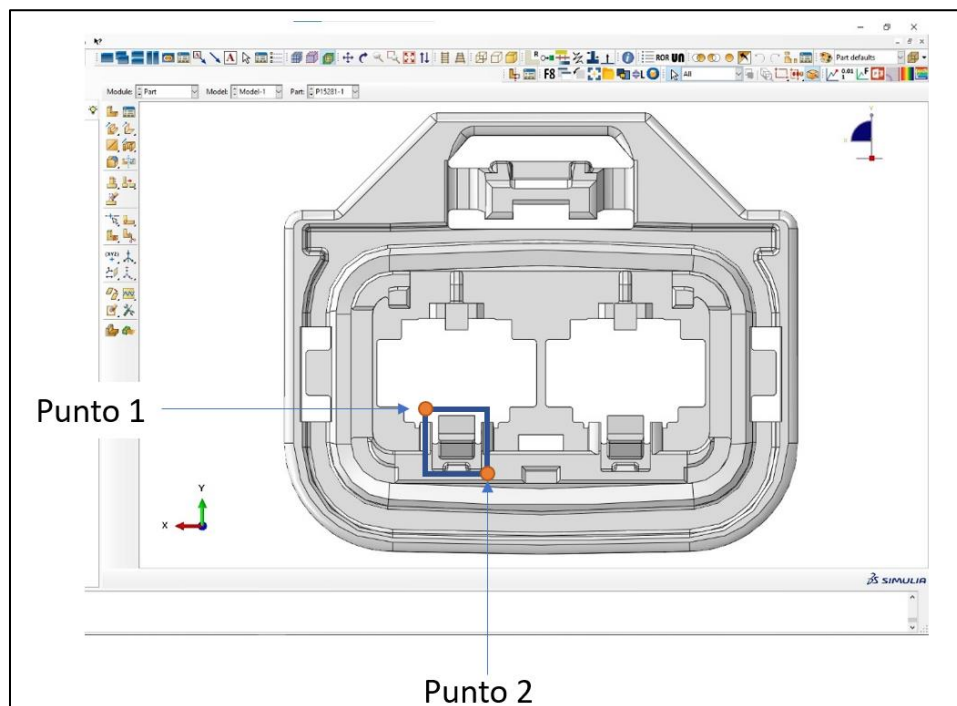


Fig 24. Traslado de rectángulo delimitador en área de trabajo de Abaqus

En la imagen de arriba representa como se traza el cuadrado delimitador en Abaqus. Con este perfil rectangular podemos realizar una extrusión que con una operación booleana de corte nos ayudara a aislar el perfil de interés del resto de la geometría

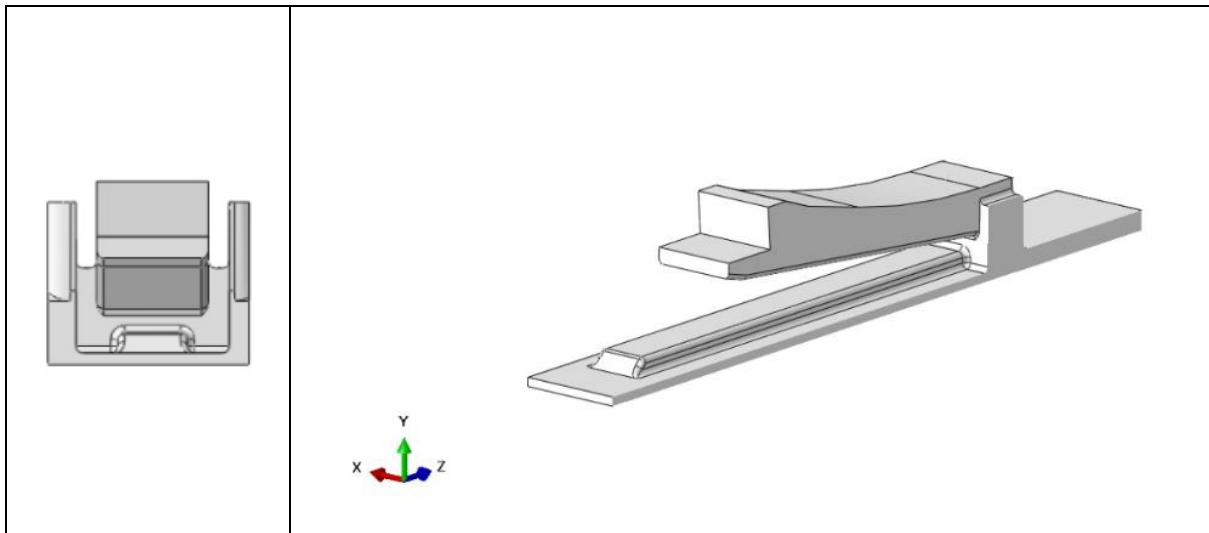


Fig. 24. Vista frontal e isométrica

La figura de la imagen anterior es la geometría resultante de nuestra simplificación, el código para realizar esto se encuentra en el anexo 1.

La implementación de esta aplicación representa un avance significativo en la utilización del software Abaqus para el análisis de elementos finitos. Al automatizar el proceso de simplificación de la geometría, no solo hemos conseguido aumentar la eficiencia y precisión de estos procedimientos, sino que también hemos mejorado la accesibilidad para los usuarios, independientemente de su nivel de experiencia previa con Abaqus.

13. Conclusiones:

Este proyecto va a permitir incrementar la productividad del departamento de elemento finito de Abaqus, que a su vez reducirá los tiempos de entrega del producto final.

El modelo entrenado no logró superar una precisión (mAP50) del 90% , esto puede ser debido a la cantidad de imágenes que se usaron para entrenar el modelo, si bien no tiene una precisión muy alta en todas las pruebas se logró al menos localizar una geometría de interés del modelo , lo que para términos de trabajo es adecuado ya que solo necesitamos analizar un Flex lock por parte y no todos estos ya que comparten la misma geometría.

Gran parte del proyecto fue consumido por la tarea de optimización, corriendo con un CPU ryzen 5 se conseguía correr un modelo yolov8n en aprox. 5 horas, por lo que al momento de la optimización teníamos que correr ese mismo modelo +100 veces para lograr una búsqueda de hiperparámetros adecuada. Un modelo de imágenes consume gran cantidad de recursos computacionales. Para culminar el proyecto se tuvo que hacer uso de una tarjeta GPU RTX con la cual se logró reducir un entrenamiento de 500 a 30 horas. Se recomienda el uso de GPU para las aplicaciones de visión computacional ya que reduce los tiempos de trabajo en gran medida.

14. Bibliography:

About Aptiv. (s. f.). Aptiv.

Our businesses Aptiv. (s. f.). Aptiv.

Jobs | Aptiv. (s. f.). Aptiv. <https://www.aptiv.com/en/jobs>

Connection Systems | Aptiv. (s. f.). Aptiv. <https://www.aptiv.com/en/solutions/connection-systems>

Ultralytics. (2024, 3 febrero). Ultralytics YOLOV8 Modes. <https://docs.ultralytics.com/es/modes/>

Ultralytics. (2024b, marzo 9). Comprehensive Tutorials to Ultralytics YOLO.

<https://docs.ultralytics.com/es/guides/>

What is W&B? | Weights & Biases Documentation. (s. f.). <https://docs.wandb.ai/guides>

Galli, S. (2022). *Python Feature Engineering Cookbook* (2.^a ed.). Packt

Publishing. <https://learning.oreilly.com/library/view/python-feature-engineering/9781804611302/>

Visengeriyeva, L., Kammer, A., Bär, I., Kniesz, A., y Plöd, M. (2023). *CRISP-ML(Q). The ML Lifecycle Process*. MLOps. INNOQ. <https://ml-ops.org/content/crisp-ml>

Piccini, N. (2023, julio 19). 101 machine learning algorithms for data science with cheat sheets. *Data Science Dojo*. <https://datasciencedojo.com/blog/machine-learning-algorithms/>

Anexo 1

Código Abaqus para simplificación de geometría

```
Proceso -x +Y

from abaqus import *
from abaqusConstants import *

#sacar el vectores de la pantalla

vv = session.viewports[session.currentViewportName].view.viewVector
cuv = session.viewports[session.currentViewportName].view.cameraUpVector

cameraUpv_I = [ i for i,x in enumerate(cuv) if x != 0.0]

vectorPantallaTemp = (vv[0]+cuv[0], vv[1]+cuv[1], vv[2]+cuv[2])

cameraSiv_I = vectorPantallaTemp.index(0) |

vv_I = [ i for i,x in enumerate(vv) if x != 0.0]

y = cuv[cameraUpv_I[0]]

xVi = cuv[cameraUpv_I[0]] * vv[vv_I[0]]

x = xVi

a = mdb.models['Model-1'].parts['P15281-1'].vertices.pointsOn

nuevaLista = [coordenadas[0] for coordenadas in a]

if xVi == 1:

    min_X = (cameraSiv_I,max(coordenada[cameraSiv_I] for coordenada in nuevaLista))
    max_X = (cameraSiv_I,min(coordenada[cameraSiv_I] for coordenada in nuevaLista))

elif xVi == -1:

    min_X = (cameraSiv_I,min(coordenada[cameraSiv_I] for coordenada in nuevaLista))
    max_X = (cameraSiv_I,max(coordenada[cameraSiv_I] for coordenada in nuevaLista))

if cuv[cameraUpv_I[0]] == 1:

    max_Y = (cameraUpv_I[0],min(coordenada[cameraUpv_I[0]] for coordenada in nuevaLista))
    min_Y = (cameraUpv_I[0],max(coordenada[cameraUpv_I[0]] for coordenada in nuevaLista))

elif cuv[cameraUpv_I[0]] == -1:

    max_Y = (cameraUpv_I[0],max(coordenada[cameraUpv_I[0]] for coordenada in nuevaLista))
    min_Y = (cameraUpv_I[0],min(coordenada[cameraUpv_I[0]] for coordenada in nuevaLista))

if vv[vv_I[0]] == 1:
```

```

elif vv[vv_I[0]] ==

    max_Z = (vv_I[0],max(coordenada[vv_I[0]] for coordenada in nuevaLista))

ESI = (min_X[1] , max_Y[1]) # esquina superior izquierda
EID = (max_X[1] , min_Y[1]) # esquina inferior izquierda

GW = max_X[1] - min_X[1] #geometry width
GH = max_Y[1] - min_Y[1] #geoemtry hight

cx = max_X[1]-((max_X[1] - min_X[1])/2.0)
cy = max_Y[1]-((max_Y[1] - min_Y[1])/2.0)

pi1 = [0,0,0]
pi2 = [0,0,0]
pi3 = [0,0,0]
pi4 = [0,0,0]

pi1[max_X[0]] = max_X[1]
pi1[max_Y[0]] = max_Y[1]
pi1[max_Z[0]] = max_Z[1]

pi2[max_X[0]] = max_X[1]
pi2[min_Y[0]] = min_Y[1]
pi2[max_Z[0]] = max_Z[1]

pi3[min_X[0]] = min_X[1]
pi3[min_Y[0]] = min_Y[1]
pi3[max_Z[0]] = max_Z[1]

pi4[min_X[0]] = min_X[1]
pi4[max_Y[0]] = max_Y[1]
pi4[max_Z[0]] = max_Z[1]

#Operacion de trazado y estrusion de corte

p = mdb.models['Model-1'].parts['P15281-1']

p = mdb.models['Model-1'].parts['P15281-1']

d2 = p.datums

d3 = d2.keys()

```

```

t = p.MakeSketchTransform(sketchPlane=d2[d3[-2]], sketchUpEdge=d2[d3[-1]],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT,
    origin=(0.0, 0.0, 0.0))

s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=96.56, gridSpacing=2.41,
    transform=t)

g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints

s1.setPrimaryObject(option=SUPERIMPOSE)

p = mdb.models['Model-1'].parts['P15281-1']

p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)

s1.rectangle(point1=(predict1[0], predict1[1]), point2=(predict2[0],predict2[1]))

p = mdb.models['Model-1'].parts['P15281-1']

d1 = p.datums

p.Wire(sketchPlane=d2[d3[-2]], sketchUpEdge=d2[d3[-1]],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, sketch=s1)

s1.unsetPrimaryObject()

del mdb.models['Model-1'].sketches['__profile__']

```