

Detección de defectos de fabricación en PCBs: estudio comparativo de tres configuraciones de YOLOv8

Luis Salvador Yábar Reaño

Facultad de Ciencias e Ingeniería

Pontificia Universidad Católica del Perú

syabar@pucp.edu.pe

Resumen

Las placas de circuito impreso (PCBs) son fundamentales en la industria electrónica, mas su fabricación exige altos estándares de calidad, ya que pequeños defectos pueden comprometer el funcionamiento de los dispositivos. Con el fin de explorar alternativas de inspección más precisas y eficientes, en este trabajo se implementaron y compararon tres variantes del modelo YOLOv8 (*nano*, *small* y *medium*) para la detección automática de defectos en PCBs. Para ello, se empleó un conjunto de datos sintético de 693 imágenes con seis tipos de defectos distintos. Los resultados muestran una mejora progresiva en las métricas a medida que aumenta la complejidad del modelo, destacando YOLOv8m como la configuración con mayor desempeño global, mientras que YOLOv8s representa un balance adecuado entre exactitud y eficiencia computacional. Estos hallazgos evidencian la viabilidad de aplicar arquitecturas modernas de *deep learning* en la inspección automatizada de PCBs.

I. INTRODUCCIÓN

Las placas de circuito impreso (PCB, por sus siglas en inglés) constituyen la base física sobre la que se montan y conectan los componentes electrónicos de prácticamente cualquier dispositivo moderno [1]. Su fabricación involucra procesos de alta precisión, ya que incluso pequeños defectos en las pistas de cobre, perforaciones o acabados pueden comprometer la funcionalidad del sistema completo. En la actualidad, todo equipo electrónico comercial incorpora una o varias PCBs, lo que convierte su producción en un proceso crítico dentro de la industria tecnológica [2].

El control de calidad en la manufactura de PCBs busca asegurar que los productos entregados estén libres de defectos. Tradicionalmente, este control se ha realizado de manera manual o mediante técnicas de inspección automatizada con reglas predefinidas, lo que puede resultar costoso y limitado frente a la complejidad creciente de los diseños. Ante esta situación, el uso de técnicas de *Deep Learning* ofrece una alternativa prometedora, capaz de detectar defectos con mayor precisión y adaptarse a variaciones en los datos [3], [4], [5].

En este contexto, el objetivo de este trabajo es implementar y comparar diferentes modelos de detección de objetos basados en la arquitectura YOLOv8 para la identificación de defectos en PCBs. Se busca evaluar el desempeño de tres variantes del modelo (*nano*, *small* y *medium*) y de esta forma determinar qué configuración ofrece el mejor balance entre exactitud y eficiencia computacional, y aportar evidencia sobre la viabilidad de aplicar este tipo de enfoques en escenarios industriales de inspección automatizada.

II. DATOS

Para abordar esta problemática se emplea un conjunto de datos de imágenes disponible en Kaggle [6]. Dicho conjunto proviene de un estudio realizado en la *Chongqing University of Technology* [4], en el cual se generó un dataset sintético y se probó una red neuronal para la detección de defectos de fabricación en PCBs.

El dataset está conformado por 693 imágenes que abarcan seis clases distintas de defectos de fabricación, con la posibilidad de que una misma imagen contenga más de un tipo de defecto. Las imágenes fueron obtenidas a partir de fotografías reales de PCBs, sobre las que se añadieron de manera artificial los defectos mediante edición por computadora.

Cada imagen se encuentra en formato .jpg, mientras que las anotaciones correspondientes a los defectos se presentan en formato VOC XML, describiendo las *bounding boxes* asociadas a cada instancia.

En cuanto a las particiones, se empleó un 80% del total de imágenes para el conjunto de entrenamiento y el 20% restante para el de prueba. Considerando la limitada cantidad de datos disponible, esta división busca maximizar el número de ejemplos de entrenamiento sin descuidar la necesidad de contar con un conjunto de prueba representativo para la evaluación del modelo.

III. METODOLOGÍA

A. Elección de ruta

La elección de la ruta para este trabajo se definió a partir de la propia naturaleza de la problemática. En un inicio, se contempló una aproximación basada en clasificación de imágenes, donde simplemente se distinguiera entre PCBs defectuosas y aquellas correctamente fabricadas. No obstante, dada la diversidad de defectos existentes, se consideró fundamental identificar no solo la presencia de fallas, sino también su tipo específico y su ubicación dentro del circuito. A partir de ello, la detección de objetos resultó ser la alternativa más adecuada. Tras investigar las opciones para detección de objetos, se eligió YOLO al ser una opción popular y viable en problemas similares [7]

B. Análisis y preprocesamiento

Como se mencionó con anterioridad, el dataset cuenta con 693 imágenes y seis clases de defectos. Los defectos considerados son: *missing hole*, *mouse bite*, *open circuit*, *short*, *spur* y *spurious copper*.

En la Tabla I se presenta un gráfico de barras con la distribución de instancias por clase. Se aprecia que, en general, la cantidad de objetos por categoría es relativamente similar. Sin embargo, la clase *spur* muestra un número considerablemente menor de ejemplos (337), mientras que la clase *spurious copper* alcanza un total de 459.

TABLE I
NÚMERO DE INSTANCIAS POR CLASE EN EL DATASET

Clase	Cantidad
missing_hole	385
mouse_bite	385
open_circuit	401
short	410
spur	337
spurious_copper	459

Cabe señalar que, en términos absolutos, el conjunto de datos es reducido, lo cual puede impactar negativamente en la capacidad de generalización de los modelos entrenados [8].

En la Figura 1 se presenta un ejemplo de cada clase de defecto, con las *bounding boxes* provistas en el dataset.

Estos defectos se caracterizan por lo siguiente [9], [10]:

- 1) **Missing hole:** Ausencia de un orificio donde debería existir una perforación en la PCB.
- 2) **Mouse bite:** Irregularidades o muescas en el borde de la placa, similares a mordidas.
- 3) **Open circuit:** Interrupción en una pista que impide la continuidad del circuito.
- 4) **Short:** Conexión no deseada entre dos pistas o pads, que genera un cortocircuito.
- 5) **Spur:** Ramificación delgada e irregular en el cobre, que puede conducir corriente de manera no intencionada.
- 6) **Spurious copper:** Presencia de fragmentos o manchas de cobre aislados que no corresponden al diseño original.

Otro aspecto fundamental a tener en consideración es el tamaño de los *bounding boxes* respecto de la imagen. Debido a que los defectos de fabricación suelen ser difíciles de apreciar a simple vista, resulta importante entender la escala en la que se encuentran. En la Figura 2 se muestra la distribución de tamaños de los *bounding boxes*.

Como se puede apreciar, los defectos de fabricación son considerablemente diminutos, ocupando en promedio un área relativa a la imagen cercana al 0.1%.

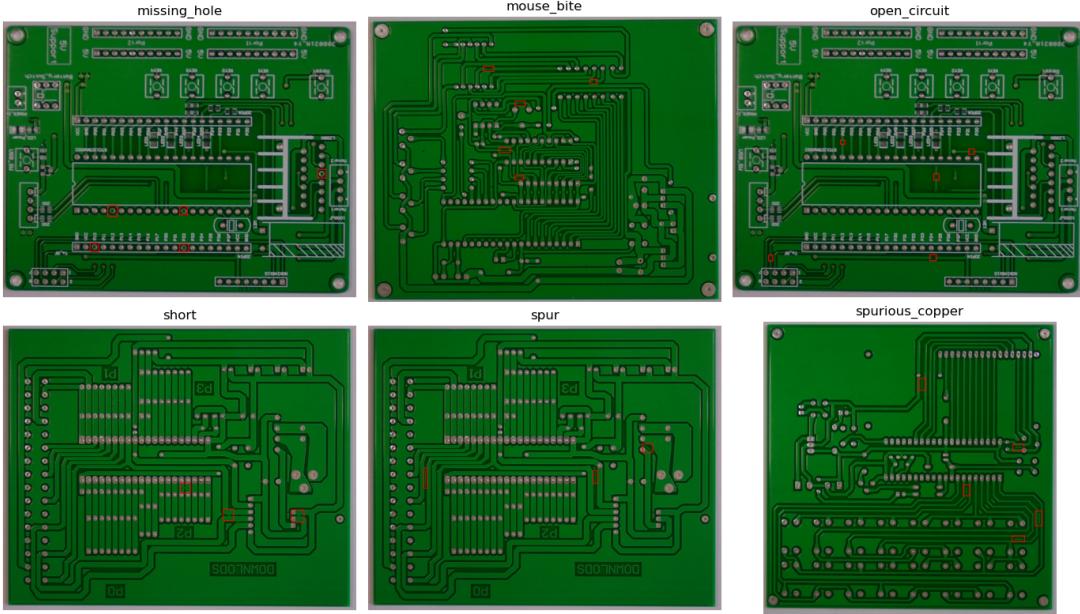


Fig. 1. Ejemplo de cada clase de defecto con sus *bounding boxes*.

En cuanto a la resolución de las imágenes, se verificó que el conjunto original presenta tamaños variables (ver Figura 3). Para el entrenamiento con YOLO se optó por un reescalado uniforme a 512×512 píxeles, lo que permite un procesamiento más eficiente y consistente en la red sin perder la información relevante de los defectos.

Respecto a las anotaciones, estas se encontraban inicialmente en formato VOC XML [11], por lo que se procedió a su conversión al formato YOLO, en el cual cada archivo .txt contiene las coordenadas normalizadas de los *bounding boxes* junto con la clase correspondiente. Finalmente, se verificó que cada imagen del conjunto cuente con su archivo de anotaciones asociado, asegurando la integridad del dataset.

Durante el entrenamiento, YOLOv8 aplica por defecto diversas transformaciones de aumento de datos (*data augmentation*) sin necesidad de configuración adicional. Entre las técnicas empleadas se destacan las siguientes:

Aumento	Descripción
hsv_h	Ajusta el tono de la imagen.
hsv_s	Modifica la saturación, afectando la intensidad de los colores.
hsv_v	Cambia el brillo de la imagen, simula diferentes condiciones de iluminación.
translate	Desplaza la imagen horizontal y verticalmente.
scale	Escala la imagen.
fliplr	Invierte la imagen de izquierda a derecha.

TABLE II

TRANSFORMACIONES DE *data augmentation* APLICADAS POR DEFECTO EN YOLOV8 (ADAPTADO DE [12])

Estos aumentos automáticos ayudan al modelo a generalizar mejor y a reducir el sobreajuste, especialmente importantes dada la reducida cantidad de datos disponibles [13].

C. Métricas

Existen diversas métricas que se suelen emplear para analizar y comparar el desempeño de modelos de detección de objetos. Para determinar las más importantes y su interpretación, se revisó [14], [15], estudios que analizan las métricas comúnmente usadas en problemas similares. A continuación se muestran las métricas consideradas en este trabajo.

- **mAP@0.5:** *Mean Average Precision* calculado con un umbral de IoU=0.5. Mide la calidad global de detección considerando que una predicción es correcta si se solapa al menos un 50% con el objeto real.

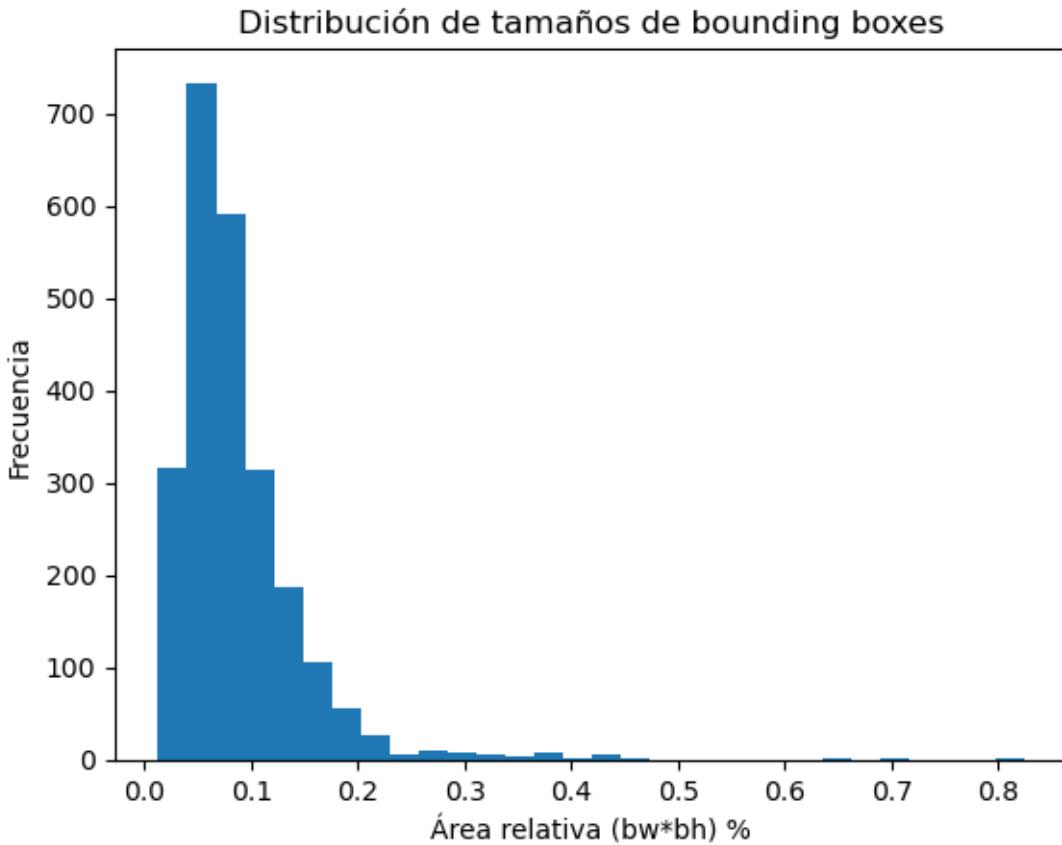


Fig. 2. Distribución de tamaños relativos de los *bounding boxes* en el dataset.

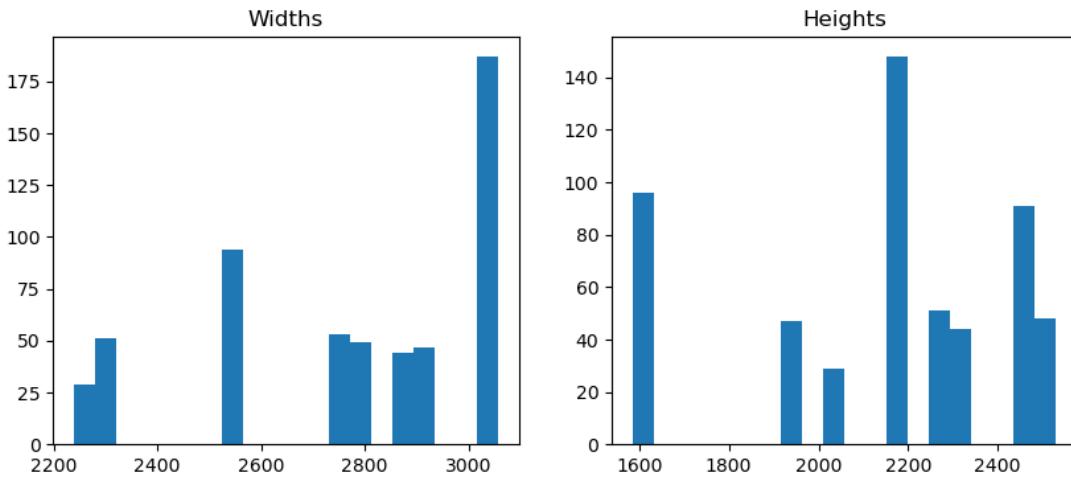


Fig. 3. Distribución de resoluciones de imágenes

- **mAP@[.5:.95]**: promedio del mAP en distintos umbrales de IoU, desde 0.5 hasta 0.95 en intervalos de 0.05. Es una métrica más estricta que refleja la precisión del detector bajo condiciones variadas de solapamiento.
- **Precision**: proporción de predicciones correctas sobre el total de predicciones realizadas. Indica qué tan confiable es el modelo cuando decide que hay un defecto, es decir, cuántos de esos realmente lo son.
- **Recall**: proporción de objetos detectados correctamente sobre el total de objetos reales. Indica qué tanto logra el modelo cubrir todos los defectos presentes, evitando falsos negativos.

D. Descripción de configuraciones

Para este trabajo se entrenaron tres variantes del modelo YOLOv8, que difieren principalmente en su tamaño, número de parámetros y capacidad de representación. A continuación, se describen sus características principales:

- **YOLOv8n (nano):** Variante más ligera, lo que permite un entrenamiento e inferencia rápidos. Su principal ventaja es la eficiencia, aunque a costa de menor desempeño en métricas de detección.
- **YOLOv8s (small):** Configuración intermedia, ofrece un equilibrio entre precisión y eficiencia, logrando mejores métricas que el modelo *nano* sin requerir la capacidad de cómputo del modelo *medium*.
- **YOLOv8m (medium):** Modelo más robusto dentro de las variantes probadas. Alcanza el mejor desempeño en términos de precisión, *recall* y mAP, a costa de un mayor consumo de recursos y tiempos de entrenamiento más prolongados.

En la Tabla III se resumen las principales características de las tres variantes de YOLOv8 empleadas en este trabajo. Se incluyen las métricas de desempeño reportadas en COCO, así como su complejidad en términos de parámetros y operaciones, lo que permite dimensionar las diferencias en precisión y costo computacional entre las configuraciones *nano*, *small* y *medium*.

TABLE III
COMPARATIVA DE CONFIGURACIONES DE YOLOV8 (DETECCIÓN, INPUT 640×640, COCO VAL2017, ADAPTADO DE [16]).

Modelo	mAP@[.5:.95]	Velocidad CPU (ms)	Velocidad GPU A100 (ms)	Parámetros (M)	FLOPs (B)
YOLOv8n	37.3	80.4	0.99	3.2	8.7
YOLOv8s	44.9	128.4	1.20	11.2	28.6
YOLOv8m	50.2	234.7	1.83	25.9	78.9

IV. RESULTADOS

A. Inferencia

Se presentan los resultados de la inferencia sobre tres imágenes seleccionadas aleatoriamente del conjunto de prueba. La primera fila corresponde al modelo *nano*, la segunda al modelo *small* y la tercera al modelo *medium*. En cada imagen se han dibujado los *bounding boxes* detectados y se ha añadido una etiqueta con la clase correspondiente, junto con el valor de confianza asociado a la predicción.

B. Gráficos por métrica y modelo

En esta sección se presentan las gráficas de evaluación correspondientes a cada uno de los modelos entrenados. La curva de *Precision-Recall* permite visualizar la relación entre la precisión y recall para distintos umbrales de confianza, mostrando el desempeño global del detector y su capacidad de mantener un equilibrio entre falsos positivos y falsos negativos.

Por otro lado, la matriz de confusión normalizada ilustra de manera detallada el desempeño por clase, indicando el porcentaje de instancias de cada categoría que fueron correctamente clasificadas y aquellos casos en los que se produjo confusión con otras clases.

1) *YOLOv8n*: La curva PR del modelo *nano* desciende de manera pronunciada, lo que evidencia que la precisión se reduce rápidamente a medida que aumenta el *recall*. En otras palabras, el modelo mantiene una buena precisión únicamente cuando detecta una menor proporción de instancias. Este patrón se repite en la mayoría de las clases, con la excepción de *missing_hole*, que presenta un comportamiento cercano al ideal, alcanzando altos valores de precisión y de *recall* al mismo tiempo.

En la matriz de confusión se observa que el modelo tiende a predecir la clase *background* de forma incorrecta en gran parte de los casos, especialmente en las clases *spur* y *mouse_bite*. De manera inversa, también se aprecia que dichas clases son predichas cuando en realidad corresponden a instancias de *background*, lo que refleja tanto falsos negativos como falsos positivos significativos.

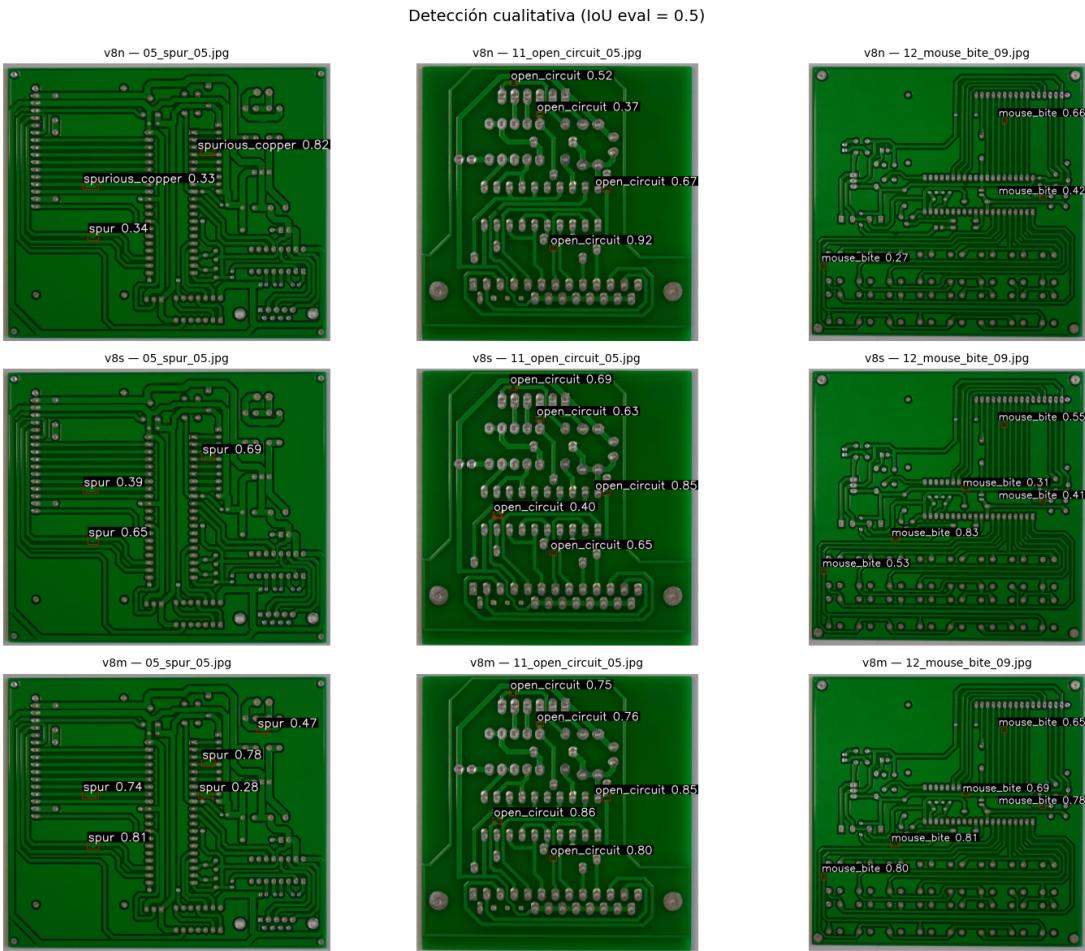


Fig. 4. Inferencia de los 3 modelos

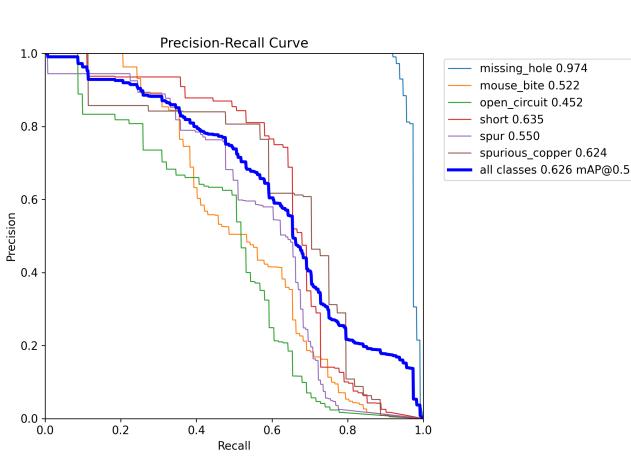


Fig. 5. Curva PR de YOLOv8n

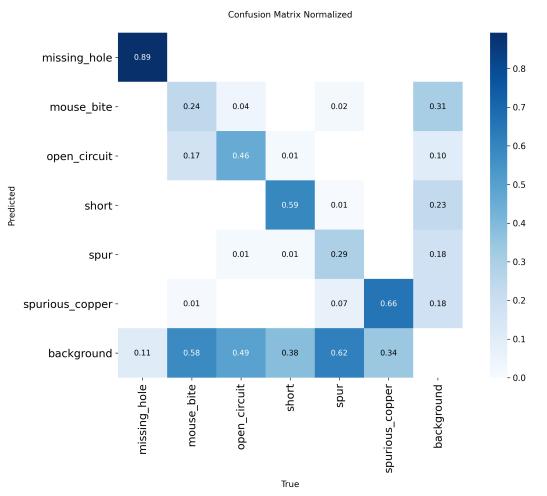


Fig. 6. Matriz de confusión normalizada de YOLOv8n

2) *YOLOv8s*: El modelo *small* muestra una mejora evidente en comparación con el modelo *nano*. La curva PR se approxima más a un comportamiento ideal, alcanzando altos niveles de precisión incluso para valores medios de *recall* (alrededor de 0.6). Como en el caso anterior, la clase *missing_hole* mantiene el mejor desempeño, aunque en este modelo las demás clases presentan resultados más cercanos a ella.

Este avance también se refleja en la matriz de confusión, donde la diagonal principal exhibe valores más altos, lo que

indica que una mayor proporción de instancias fueron clasificadas correctamente. No obstante, el modelo continúa mostrando confusiones con la clase *background*, generando falsos positivos y falsos negativos, aunque en una magnitud menor que en el modelo *nano*.

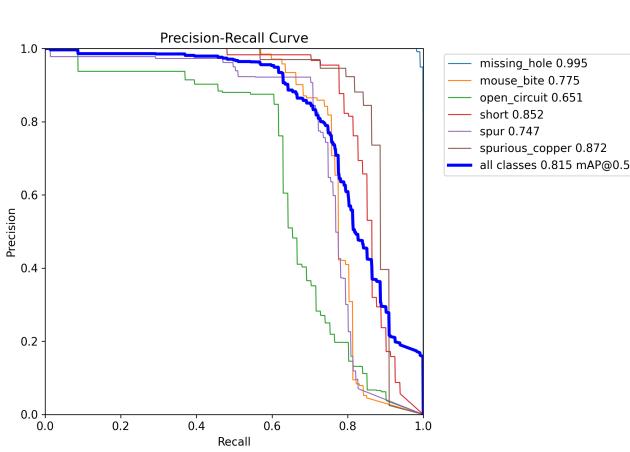


Fig. 7. Curva PR de YOLOv8s

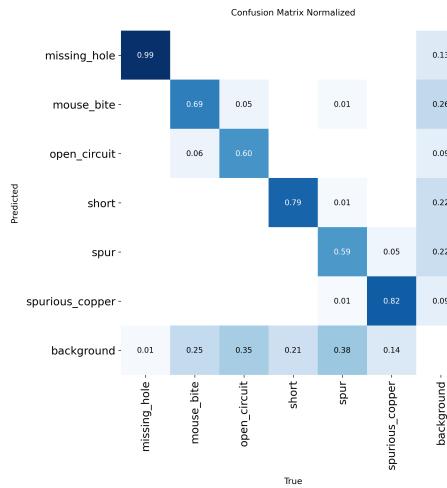


Fig. 8. Matriz de confusión normalizada de YOLOv8s

3) *YOLOv8m*: Finalmente, el modelo *medium* alcanza el mejor desempeño, con un mAP@0.5 de 0.896 considerando todas las clases, y un comportamiento mucho más cercano al ideal en las curvas PR. Todas las clases superan una precisión de 0.8, e incluso algunas sobrepasan el 0.9.

Este rendimiento se refleja en una mayor proporción de objetos correctamente detectados, visible en los altos valores de la diagonal principal de la matriz de confusión. Es resaltable el 100% de precisión y recall en la clase *missing_hole*, la cual ya venía teniendo un buen desempeño con los demás modelos. Si bien persisten ciertas confusiones con la clase *background*, estas se reducen de manera considerable en comparación con los otros modelos: los falsos positivos se ubican por debajo de 0.31, mientras que los falsos negativos se reducen a valores inferiores a 0.25.

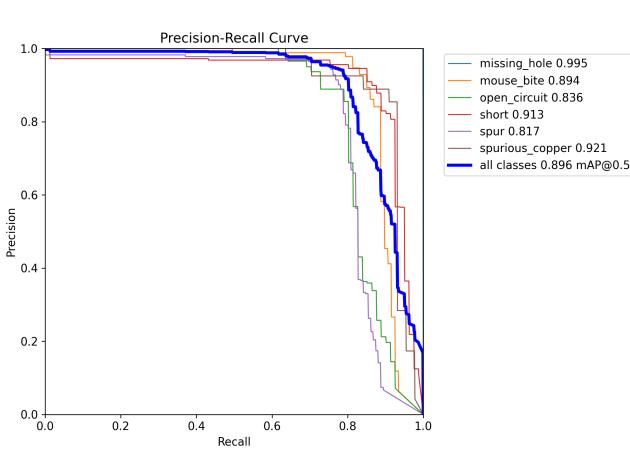


Fig. 9. Curva PR de YOLOv8m

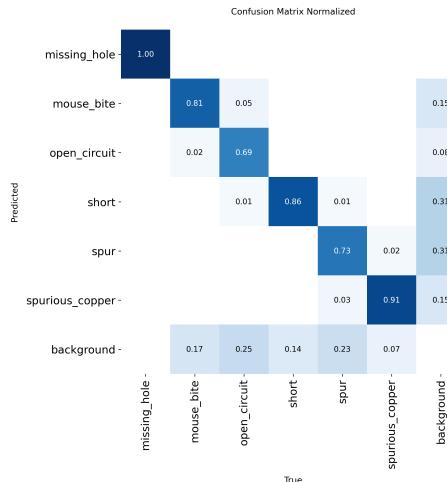


Fig. 10. Matriz de confusión normalizada de YOLOv8m

C. Comparativa

En esta sección se comparan directamente los tres modelos de YOLO presentados con anterioridad.

La Tabla IV presenta un resumen de la comparación entre modelos y sus métricas obtenidas en el entrenamiento y prueba.

TABLE IV
RESULTADOS DE ENTRENAMIENTO Y PRUEBA PARA CADA MODELO.

Model	Precision_train	Recall_train	mAP@0.5_train	Precision_test	Recall_test	mAP@0.5_test
v8n	0.7363	0.5692	0.6276	0.7333	0.5694	0.6263
v8s	0.8543	0.7647	0.8150	0.8876	0.7555	0.8152
v8m	0.9318	0.8472	0.8972	0.9383	0.8314	0.8959

La Figura 11 muestra tres comparaciones de los parámetros de los modelos, facilitando la visualización de la diferencia en el desempeño de los mismos.

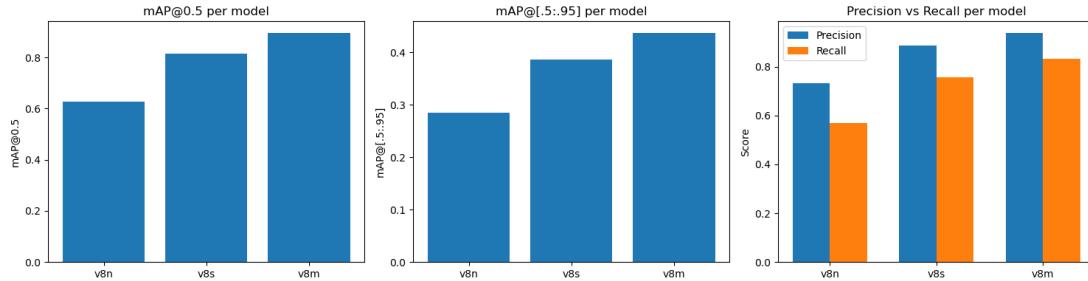


Fig. 11. Comparación de los 3 modelos

A partir de lo anterior, es claro el dominio del modelo 3 sobre los otros dos, en todas las métricas evaluadas. Este resultado era esperable, ya que cada modelo era más complejo y pesado computacionalmente que el anterior.

V. CONCLUSIONES

A. Elección de configuración

El modelo *medium* obtuvo el mejor desempeño en todas las métricas evaluadas, por lo que constituye la elección final más adecuada para esta problemática. No obstante, cabe resaltar el rendimiento alcanzado por el modelo *small*, que si bien presenta resultados inferiores, ofrece la ventaja de ser considerablemente más ligero en términos computacionales. En consecuencia, la elección entre ambas variantes puede depender de los recursos disponibles: cuando se disponga de capacidad de cómputo suficiente conviene utilizar el modelo *medium*, mientras que en entornos con mayores restricciones podría optarse por el modelo *small*, aceptando una ligera pérdida de desempeño.

B. Lecciones aprendidas

A lo largo del proyecto se evidenció la utilidad de las curvas *Precision-Recall* y de las matrices de confusión para analizar y comparar el desempeño de diferentes modelos, logrando una comprensión más clara de sus fortalezas y debilidades. Asimismo, se constató la relevancia del análisis exploratorio de datos para explicar los resultados obtenidos y reconocer las limitaciones inherentes a los modelos empleados. Durante el proceso de entrenamiento también se hizo evidente el costo en tiempo y recursos que implica entrenar redes profundas, lo cual se convierte en un factor crítico en problemas con conjuntos de datos más grandes, imágenes de mayor resolución o entrenamientos prolongados con más épocas.

VI. LIMITACIONES Y CONSIDERACIONES ÉTICAS

La principal limitación de este estudio es la reducida cantidad de ejemplos en el conjunto de datos. Incluso con modelos potentes, la falta de diversidad en los datos limita su capacidad de generalización y, por ende, el nivel de desempeño alcanzable. Adicionalmente, al tratarse de datos sintéticos, existe el riesgo de que no reflejen con fidelidad las condiciones de producción reales, lo que podría comprometer la eficacia del modelo en un entorno industrial. Por este motivo, antes de plantear un despliegue sería indispensable realizar pruebas con imágenes reales, recolectadas en distintas fábricas y bajo diferentes condiciones de operación.

Desde una perspectiva ética, resulta fundamental validar rigurosamente que el modelo es capaz de detectar defectos con suficiente seguridad en escenarios reales. La presencia de falsos positivos o falsos negativos podría no solo ocasionar pérdidas económicas, sino también llevar al descarte masivo de placas de circuito impreso que, en caso de no ser recicladas adecuadamente, constituirían un problema ambiental significativo.

REFERENCES

- [1] Robert Keim, “What Is a Printed Circuit Board (PCB)?” <https://www.allaboutcircuits.com/technical-articles/what-is-a-printed-circuit-board-pcb>, Apr. 2020.
- [2] “The Global Printed Circuit Board Market: Key Insights, Forecasts, & Growth Outlook,” <https://www.mktpcb.com/pcb-industry-statistics-trends-infographic/#:~:text=PCB%20Market%20Size%2C%20CAGR%2C%20and,3.3%25%20during%202021-2026.,> 2023.
- [3] B. Hu and J. Wang, “Detection of PCB Surface Defects With Improved Faster-RCNN and Feature Pyramid Network,” *IEEE Access*, vol. 8, pp. 108 335–108 345, 2020.
- [4] R. Ding, L. Dai, G. Li, and H. Liu, “TDD-net: A tiny defect detection network for printed circuit boards,” *CAAI Transactions on Intelligence Technology*, vol. 4, no. 2, pp. 110–116, Jun. 2019.
- [5] Q. Ling and N. A. M. Isa, “Printed Circuit Board Defect Detection Methods Based on Image Processing, Machine Learning and Deep Learning: A Survey,” *IEEE Access*, vol. 11, pp. 15 921–15 944, 2023.
- [6] Alfiia Akhatova, “PCB Defects,” <https://www.kaggle.com/datasets/akhatova/pcb-defects>, 2021.
- [7] M. Hussain, “YOLOv5, YOLOv8 and YOLOv10: The Go-To Detectors for Real-time Vision,” Jul. 2024.
- [8] Shubham Patni, “Generalisation, Training-Validation & Test data. Machine Learning- Part 6.” <https://medium.com/@shubhapatnim86/generalisation-training-validation-test-data-machine-learning-part-6-1de9dbb7d3d5>, May 2018.
- [9] Tara Dunn, “Common PCB Assembly Defects You Should Know,” <https://resources.altium.com/p/common-pcb-assembly-defects-you-should-know>, Jul. 2024.
- [10] D. S. Koblah, O. P. Dizon-Paradis, J. Schubeck, U. J. Botero, D. L. Woodard, and D. Forte, “A Comprehensive Taxonomy of Visual Printed Circuit Board Defects,” *Journal of Hardware and Systems Security*, vol. 7, no. 2-3, pp. 25–43, Sep. 2023.
- [11] Roboflow, “Pascal VOC XML,” <https://roboflow.com/formats/pascal-voc-xml>, 2021.
- [12] Ultralytics, “Model Training with Ultralytics YOLO,” <https://docs.ultralytics.com/modes/train/#train-settings>, Nov. 2023.
- [13] I. D. Fofanah, *7 Effective Ways to Deal with a Small Dataset*, Aug. 2022.
- [14] R. Padilla, S. L. Netto, and E. A. B. Da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. Niterói, Brazil: IEEE, Jul. 2020, pp. 237–242.
- [15] Henrique Vedoveli, “Metrics Matter: A Deep Dive into Object Detection Evaluation,” <https://medium.com/@henriquevedoveli/metrics-matter-a-deep-dive-into-object-detection-evaluation-ef01385ec62>, Sep. 2023.
- [16] Ultralytics, “Ultralytics YOLOv8 Overview,” <https://docs.ultralytics.com/models/yolov8/>, Nov. 2023.