

Reconocimiento de componentes electrónicos con CNN

Profesor: Leandro Borgnino

Francesco Gallone



Fundación Tarpuy - Fulgor

Índice

1. Introducción	2
1.1. Motivación	2
1.2. Objetivo	2
2. Marco Teórico	3
2.1. Principio de Operación	3
2.2. Propiedades de las <i>CNN</i>	3
2.3. Comparativa entre <i>CNN</i> y <i>ANN</i>	3
2.4. Componentes Principales de las <i>CNN</i>	4
2.5. Arquitecturas clásicas de <i>CNN</i>	5
2.6. Aplicaciones en procesamiento de imágenes	5
2.7. ¿Para que sirven las <i>redes neuronales convolucionales</i> ?	5
3. Implementación	7
4. Resultados	9
4.1. Etiquetas verdaderas VS Etiquetas predichas	10
4.2. Pruebas con imágenes ajenas al dataset	14
5. Conclusiones	16

1. Introducción

1.1. Motivación

La detección y clasificación de objetos es un área clave de la visión por computadora, que permite a las máquinas identificar y clasificar objetos dentro de imágenes o videos. Esta tecnología tiene aplicaciones en sectores como la automoción, la seguridad, la medicina y la industria, donde su uso ha mejorado la eficiencia y automatizado numerosos procesos.

En la industria, se utiliza para tareas como el control de calidad y la clasificación de productos, lo que optimiza la productividad y reduce errores. En el ámbito de la seguridad, los sistemas de videovigilancia basados en visión por computadora permiten detectar comportamientos sospechosos y reconocer rostros en tiempo real, lo que mejora la prevención y respuesta ante incidentes.

En los vehículos autónomos, la detección de objetos es esencial para identificar otros autos, peatones y señales de tránsito, permitiendo la conducción autónoma de manera segura. Además, en la medicina, se aplica para el análisis de imágenes médicas, ayudando a identificar tumores, fracturas y otras anomalías, lo que facilita diagnósticos más rápidos y precisos. En la agricultura, la visión por computadora ayuda a detectar plagas o enfermedades en los cultivos, mejorando la eficiencia en el uso de recursos y aumentando la producción.

En resumen, la detección y clasificación de objetos está transformando diversas áreas, mejorando la automatización de tareas, optimizando procesos y ofreciendo soluciones más eficientes y seguras.

1.2. Objetivo

Este proyecto tiene como enfoque el reconocimiento de varios componentes electrónicos de tipo montaje superficial en distintos PCB de forma simultánea, el modelo mostrará que tan seguro está sobre la identidad de cada componente detectado y utiliza el dataset PCB Computer Vision Project proveniente de RoboFlow. Como modelo se utiliza YoloV8 ya que ofrece un balance ideal entre velocidad, precisión y simplicidad, esto último lo vuelve muy fácil de usar, ahora específicamente se utiliza la versión V8 ya que mejora la precisión de múltiples objetos en la misma imagen (ideal para este proyecto), es más rápido, tiene detección sin anclas mejorando la detección de objetos de distinto tamaño (como se puede llegar a encontrar en un PCB) y por último es menos propenso al sobre-ajuste y generaliza mejor en datos no vistos, ideal para los dataset que contengan errores inusuales (más adelante veremos que este dataset es uno de ellos)

2. Marco Teórico

2.1. Principio de Operación

La convolución es la operación fundamental de las CNN y se utiliza para aplicar un conjunto de filtros entrenables sobre la entrada. Un filtro es una pequeña matriz de pesos que se desliza sobre la imagen de entrada, multiplicando sus valores con los correspondientes en la matriz del filtro y sumando los resultados. Esto genera un "mapa de características", que resalta patrones específicos como bordes, texturas, o formas.

La salida de un filtro aplicado a una región de la imagen se calcula como:

$$z_{j,k} = \sigma \left(\sum_{l=0}^n \sum_{m=0}^n W_{l,m} \cdot X_{j+l,k+m} + b \right) \quad (1)$$

W : Matriz de pesos del filtro

X : Valores de la imagen en la región actual

b : sesgo

σ : función de activación no lineal

2.2. Propiedades de las *CNN*

-Interacciones dispersas: Cada neurona de salida está conectada solo a un subconjunto de neuronas de entrada, lo que reduce significativamente el número de cálculos necesarios en comparación con las ANN

-Uso Compartido de Parámetros: Los pesos del filtro son los mismos para todas las posiciones de la imagen, lo que reduce el número total de parámetros del modelo y permite que la red sea más eficiente.

-Invariación a traslaciones: Si un objeto en la imagen se desplaza (por sacar la imagen "corrida"), la salida del modelo también se desplaza de manera proporcional, permitiendo identificar patrones independizandolos de la posición.

2.3. Comparativa entre *CNN* y *ANN*

Uno de los principales beneficios de las CNN es su mayor eficiencia computacional. En una ANN, para calcular las conexiones entre M entradas y N salidas, la complejidad es $O(MXN)$. En cambio, en las CNN, la complejidad depende del tamaño del filtro K , resultando en $O(KXN)$. Dado que K suele ser mucho menor que M , las CNN son más eficientes, especialmente para imágenes grandes.

2.4. Componentes Principales de las CNN

Capas convolucionales

-Estas capas aplican múltiples filtros a las imágenes de entrada para generar mapas de características. Cada filtro aprende a detectar un patrón específico, como bordes, texturas o regiones de color.

Pooling

La capa de pooling reduce las dimensiones espaciales de los mapas de características. Los métodos más comunes son:

Max Pooling: selecciona el valor máximo en una región específica.

Average Pooling: Calcula el promedio de los valores en una región.

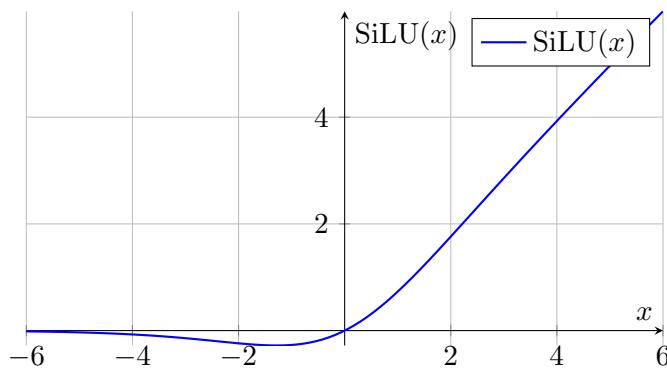
Funciones de activación

Introducen no linealidad en el modelo, permitiendo que las CNN aprendan relaciones complejas. La más utilizada es ReLU (*Rectified Linear Unit*), que se define como:

$$f(x) = \max(0, x) \quad (2)$$

Ecuación de la función de activación ReLU.

En este caso se utiliza SiLu como función de activación porque tiene como ventaja mejor manejo de pequeñas activaciones ya que es suave y no anula valores pequeños como ReLU (que los convierte en 0), ayudando a capturar características más sutiles, particularmente ideal para este modelo ya que por ejemplo la diferencia entre capacitores de montaje superficial y resistencias en algunos casos es mínima. Podemos ver esta suavidad en su función.



$$\text{SiLU}(x) = x \cdot \frac{1}{1 + e^{-x}} \quad (3)$$

Ecuación de la función de activación SiLu.

Normalización

Se utilizan técnicas como Batch Normalization para estabilizar y acelerar el entrenamiento ajustando la distribución de las activaciones en cada capa.

Capas completamente conectadas

Estas capas conectan todas las neuronas de entrada con las de salida, permitiendo que la red combine las características extraídas para producir una decisión final

2.5. Arquitecturas clásicas de *CNN*

LeNet-5

Introducida en 1998, fue una de las primeras CNN exitosas, utilizada para el reconocimiento de dígitos escritos a mano. Consistía en 2 capas convolucionales y 3 capas completamente conectadas.

AlexNet

Revivió el entusiasmo en el campo de la inteligencia artificial en 2012 al ganar el concurso ImageNet, usando 5 capas convolucionales y 3 completamente conectadas. Introdujo el uso extensivo de GPUs para acelerar el entrenamiento.

VGG-16

Propuso un diseño profundo con 13 capas convolucionales y 3 capas completamente conectadas, estandarizando el uso de filtros de tamaño 3X3.

ResNet

Introdujo bloques residuales que permitieron entrenar redes muy profundas al mitigar problemas de gradientes vanishing.

2.6. Aplicaciones en procesamiento de imágenes

Las CNN demostraron ser herramientas versátiles y efectivas para una amplia gama de aplicaciones, algunas como:

- Clasificación de imágenes: identificar la clase de un objeto en una imagen.
- Detección de objetos: localizar y clasificar múltiples objetos dentro de una imagen.
- Segmentación semántica: asignar etiquetas a cada píxel de una imagen.
- Inspección visual: identificar defectos o anomalías en componentes, como placas de circuitos impresos (PCBs).

2.7. ¿Para qué sirven las *redes neuronales convolucionales*?

Las **redes neuronales convolucionales (CNNs)** tienen un uso muy extendido en distintos campos gracias a su capacidad para procesar y analizar datos estructurados, como imágenes y videos. En visión por computadora, resultan de gran importancia para tareas como clasificación de imágenes, detección de objetos, segmentación de escenas y reconocimiento facial. En el área de la salud, se utilizan para interpretar imágenes diagnósticas, detectar tumores y delimitar órganos, aportando herramientas valiosas en el diagnóstico médico, este último comienza a verse cada vez más ya que antes se veía con escepticismo y se evitaba.

En el procesamiento de lenguaje natural, las CNNs se aplican para clasificar textos y extraer información relevante, mientras que en finanzas y negocios son útiles para analizar series de datos y documentos digitalizados. En robótica, facilitan la percepción del entorno y la planificación de movimientos, haciendo posible que los sistemas autónomos operen con mayor precisión. Además,

en ciencia y tecnología se emplean para identificar galaxias en astronomía o analizar imágenes satelitales en estudios geográficos y ambientales.

Estas redes se diferencian de las redes neuronales artificiales tradicionales (ANN) al aprovechar la operación de convolución en lugar de la multiplicación matricial general para procesar información estructurada espacialmente. Este enfoque permite extraer características locales relevantes mientras se conserva la información espacial de las entradas, como imágenes o señales visuales.

El principal motivo del desarrollo de las CNN es **superar las limitaciones** de las ANN en tareas visuales, ya que:

- Las ANN requieren un número masivo de parámetros para modelar relaciones complejas, lo que dificulta su entrenamiento.

- No pueden preservar la espacialidad inherente de los datos, perdiendo relaciones entre píxeles vecinos.

- No son invariantes a traslaciones, es decir, el modelo puede fallar si un objeto en la imagen se mueve ligeramente.

3. Implementación

El dataset ***pcb Computer Vision Project*** cuenta con un total de 6265 imágenes en total dividiéndose así en 4384 para entrenamiento, 1235 para validación y 621 para testeo, cada una con sus respectivas etiquetas. A su vez, por clasificación de componentes se pueden dividir en 10 categorías distintas (bead, capacitor, connector, crystal, ic, inductor, led, resistor, semiconductor, switch), de las cuales por un mal diseño del dataset se utilizaron 8 (ya que 1 de ellas (bead) tenía menos de 5 imágenes y la otra (inductor) tenía menos de 10) provocando un sesgo a la hora de obtener resultados finales, ya que hay casi nula densidad de imágenes para esas clases provocando que pueda confundirse con demás componentes.

El modelo elegido en un principio fue YoloV5, pero los resultados arrojados por este modelo fueron tan insatisfactorios que se evitó agregar ya que no tiene caso compararlo, con el que fue elegido posteriormente el cual fue YoloV8. Este fue elegido por los argumentos expresados con anterioridad y se compone de 224 capas, a fines de evitar confusión en el lector, se agrupó los tipos de capas en cantidades para no sobre-extender la proyección de las mismas

Tipo de Capa	Cantidad	Parámetros Totales
DetectionModel	1	3,012,798
Sequential	7	3,766,044
Conv	57	2,998,352
Conv2d	64	3,002,398
BatchNorm2d	57	10,400
SiLU	1	0
C2f	8	1,517,120
ModuleList	10	1,700,382
Bottleneck	10	947,136
SPPF	1	164,608
MaxPool2d	1	0
Upsample	2	0
Concat	4	0
Detect	1	753,262
DFL	1	16

Figura 1: *Lista de capas agrupadas respecto a la cantidad de cada una.*

Para entender como funciona el paso a paso de Yolo entre capa y capa podemos ver el siguiente diagrama. Como se puede notar no aparecen todas las capas de la figura de lista de capas pero, básicamente, así es como funciona de forma resumida, ya que la mayoría de las veces se repite este ciclo. Si está interesado en ver el paso a paso de las 224 capas puede verlo en este enlace.

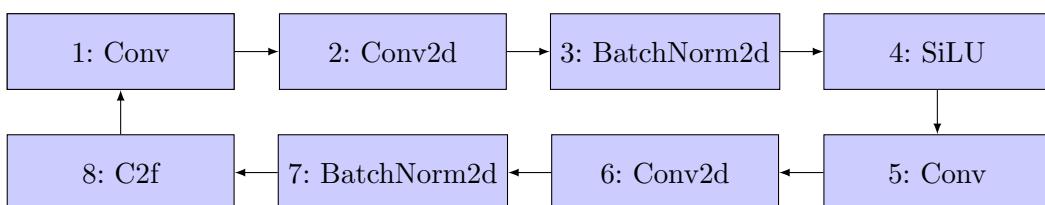


Diagrama conceptual resumido del funcionamiento de YoloV8.

Este modelo utiliza la función de optimización *AdamW* que tiene un equilibrio entre velocidad de convergencia y la generalización del modelo. Se le dieron 25 épocas como parámetro a Yolo, con una dimensión de 640x640 (usada normalmente en este tipo de modelos).

4. Resultados

Como se aclaró antes este dataset se compone de 10 clases de componentes de las cuales 2 de ellas tenían 3 y 8 imágenes, algo muy pobre para los requerimientos de entrenamiento por lo que bajaron considerablemente el rendimiento general del modelo (0.64), provocando un sesgo que hace pensar que el modelo es de bajo rendimiento.

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95:
all	1235	40022	0.64	0.388	0.466	0.344
bead	3	7	0	0	0	0
capacitor	884	15987	0.912	0.333	0.461	0.282
connector	1032	5259	0.843	0.438	0.575	0.432
crystal	28	30	0.827	0.733	0.807	0.72
ic	702	3161	0.856	0.417	0.572	0.417
inductor	8	25	0	0	0.0458	0.0294
led	425	1577	0.917	0.377	0.466	0.321
resistor	861	13822	0.876	0.306	0.417	0.244
semiconductor	37	81	0.551	0.484	0.549	0.445
switch	11	73	0.616	0.795	0.763	0.546

Figura 2: Métricas del modelo

Tal como se ve en la imagen las clases que tienen más imágenes son las que más precisión tienen. Por lo que en adelante para evitar este sesgo se excluye a estas dos clases. Ahora si vemos como funciona el modelo sin estas dos clases veremos como el rendimiento es considerablemente mayor.

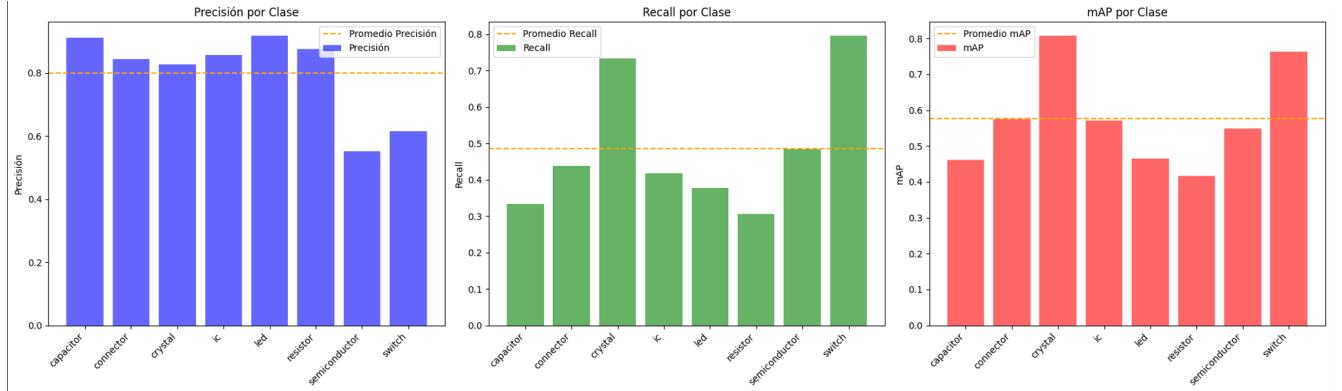


Figura 3: Métricas por clase

Precisión: Mide la proporción de predicciones correctas entre todas las predicciones hechas por el modelo.

Recall (sensibilidad): Mide la capacidad del modelo para detectar los objetos presentes en la imagen. Un alto recall significa que el modelo detecta la mayoría de los objetos.

mAP@50 (Mean Average Precision at IoU 0.5): Es el promedio de la precisión para todas las clases, usando un umbral de IoU de 0.5, IoU (*Intersection over Union*) es una métrica utilizada para medir la superposición entre dos cajas delimitadoras, una predicha por el modelo y otra real, un IoU del 0.5 significa que la intersección entre la caja predicha y la caja real cubre al menos un 50 por ciento de la unión.

Específicamente para este modelo de detección de componentes electrónicos es difícil obtener un buen mAP50 o mAP50-95 ya que las dimensiones de los objetos a detectar son muy

pequeñas (la mayoria son de montaje superficial) y se repiten mucho en un pcb que puede ser grande, ademas los problemas en iluminación y la similitud entre algunos componentes sumado a que los componentes pueden variar en su orientación y no necesariamente estar horizontal o verticalmente dificultando aún más la superposición de cajas.

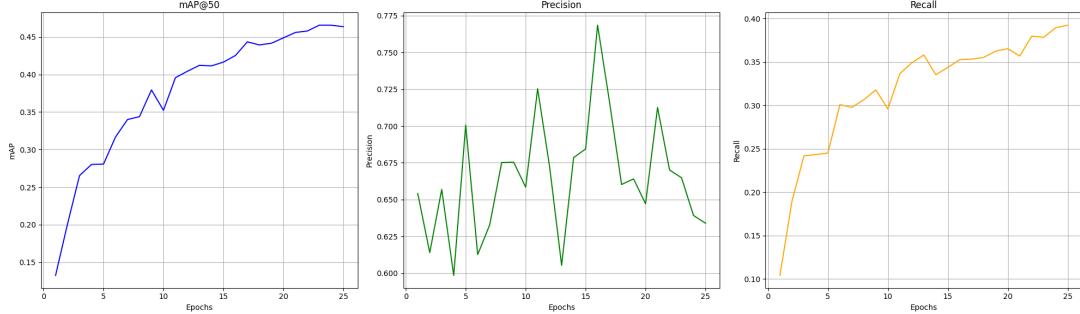


Figura 4: *Métricas por época*

Como se ve en la figura anterior el recall y mAP@50 (se cumple tambien para el mAP@50-95) aumentan conforme se le agregan más épocas, la precisión en cambio puede variar ya sea por overfitting o underfitting. Estas métricas por época se hicieron juntando a todas las clases por lo que probablemente estos bajos niveles se deben a las clases defectuosas del dataset.

4.1. Etiquetas verdaderas VS Etiquetas predichas

Ahora podemos hablar de los resultados del modelo y sacar conclusiones sobre si realmente es un buen modelo o tiene bajo rendimiento como el que se obtuvo con el promedio de todas las clases. Para todas las imagenes, en el lado izquierdo estarán las etiquetas verdaderas y en el derecho las predichas. A las etiquetas predichas se les agregó un porcentaje para saber que tan seguro está el modelo de la identificación de ese objeto y se usan abreviaciones para cada clase asi se mitiga (pero no se logra eliminar del todo) la superposición de texto de las etiquetas. Las abreviaciones cumplen lo siguiente:

bead = bd

capacitor = cap

connector = con

crystal = crys

ic = ic

inductor = ind

led = led

resistor = res

semiconductor = semi

switch = sw

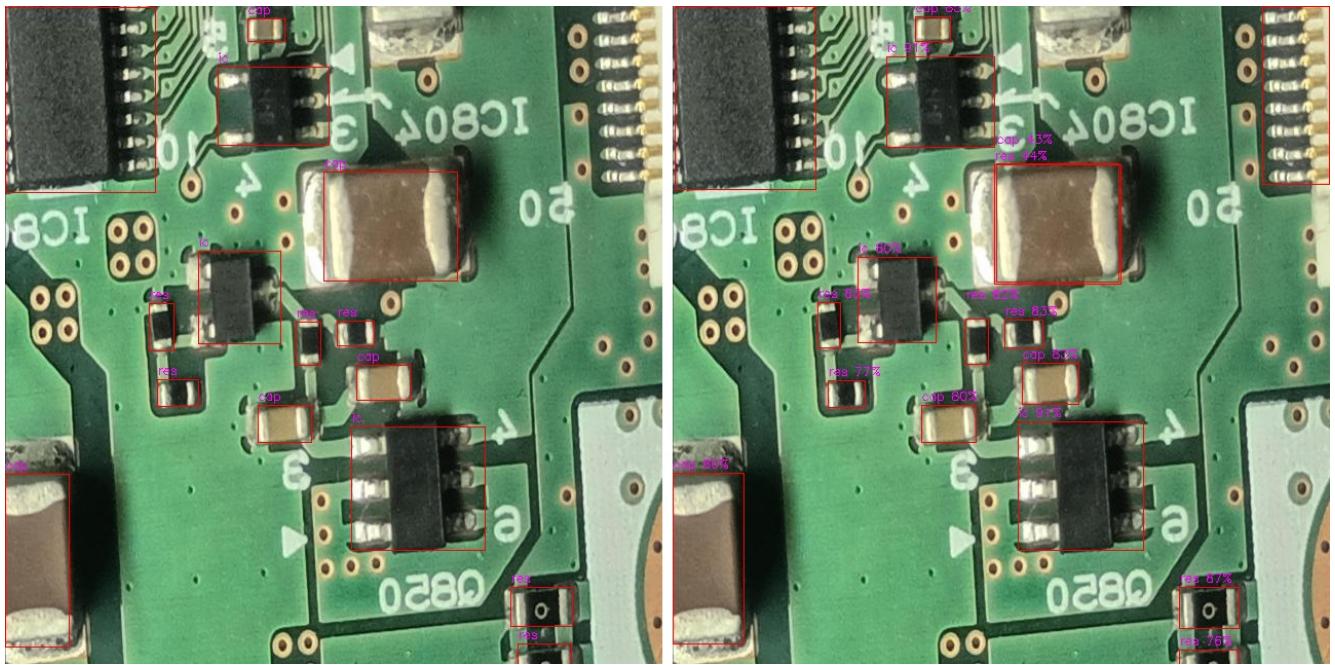


Figura 5: Verdadero vs Predicho

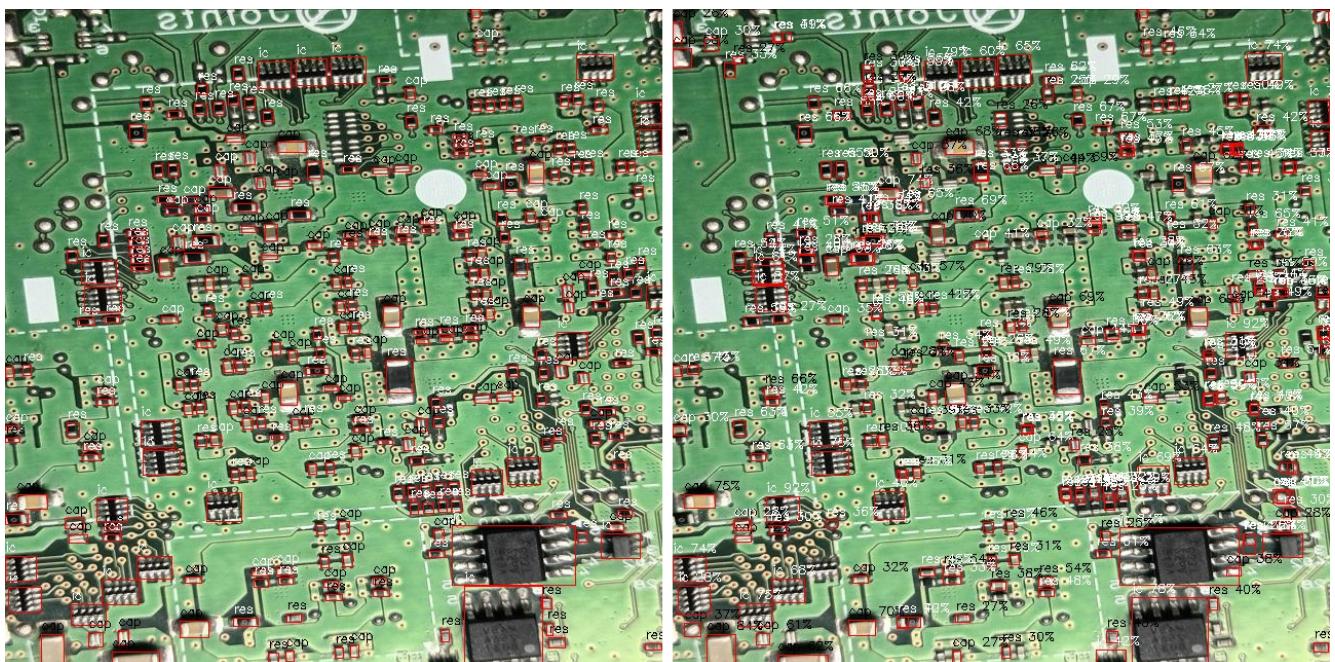


Figura 6: Verdadero vs Predicho

Probablemente sea difícil la correcta lectura en una imagen con tantos objetos pequeños por lo que se debe colocar a las etiquetas colores que contrasten bien el fondo del PCB

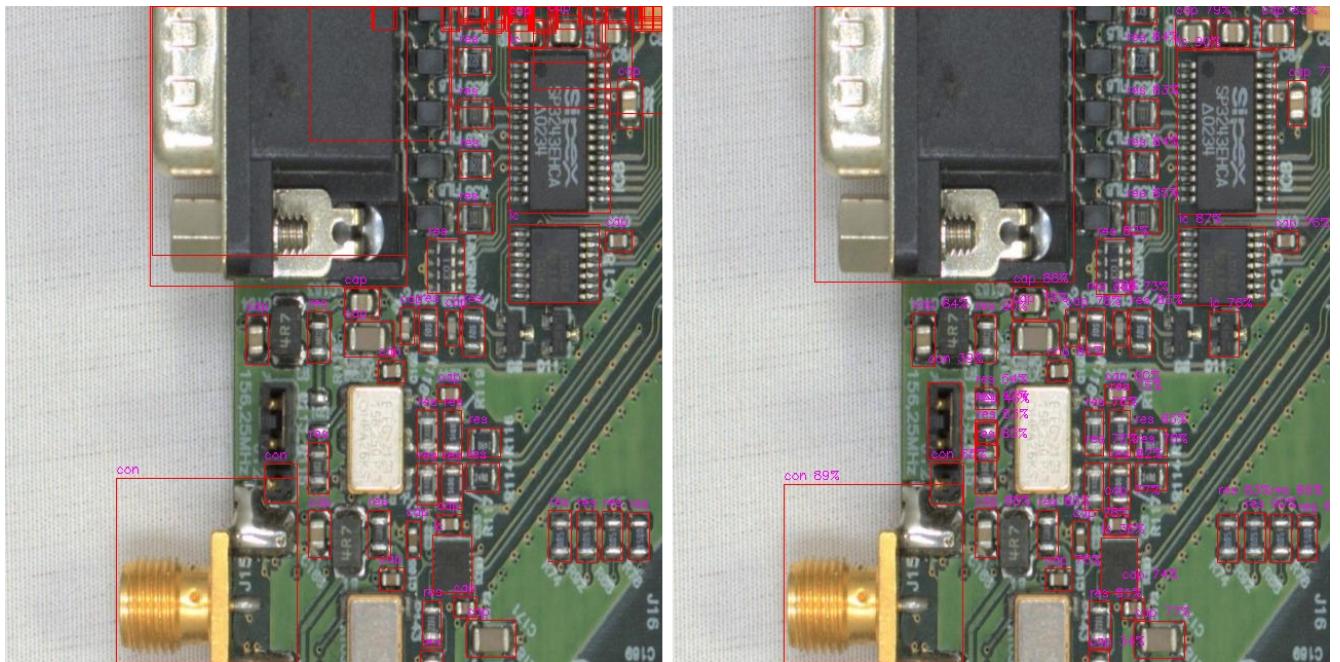


Figura 7: Verdadero vs Predicho

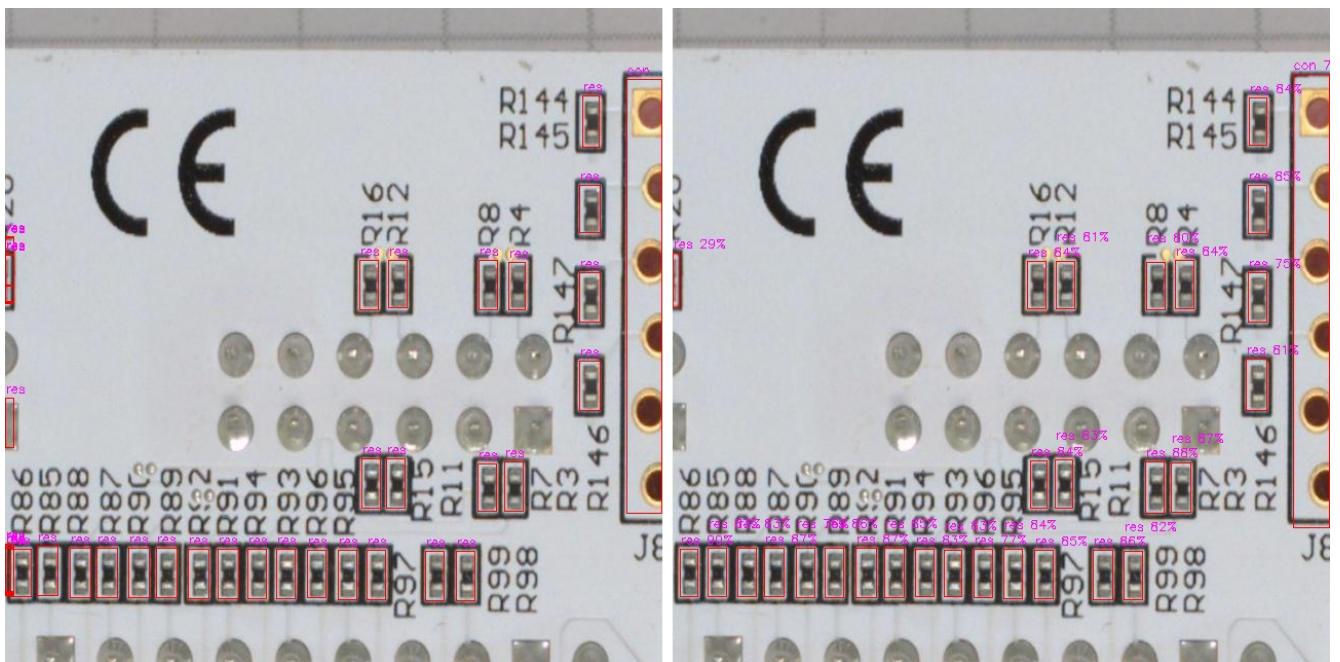


Figura 8: Verdadero vs Predicho

Sin tener la certeza, se estima que el dataset usado utilizo algun script para generar las etiquetas ya que en varias imagenes se puede ver como hay muchas etiquetas verdaderas superpuestas en alguna esquina de la imagen haciendo que el modelo creado con Yolov8 tenga falsos negativos bajandole artificialmente la eficacia por lo que se sugiere obtener las métricas de forma manual en cada imagen.

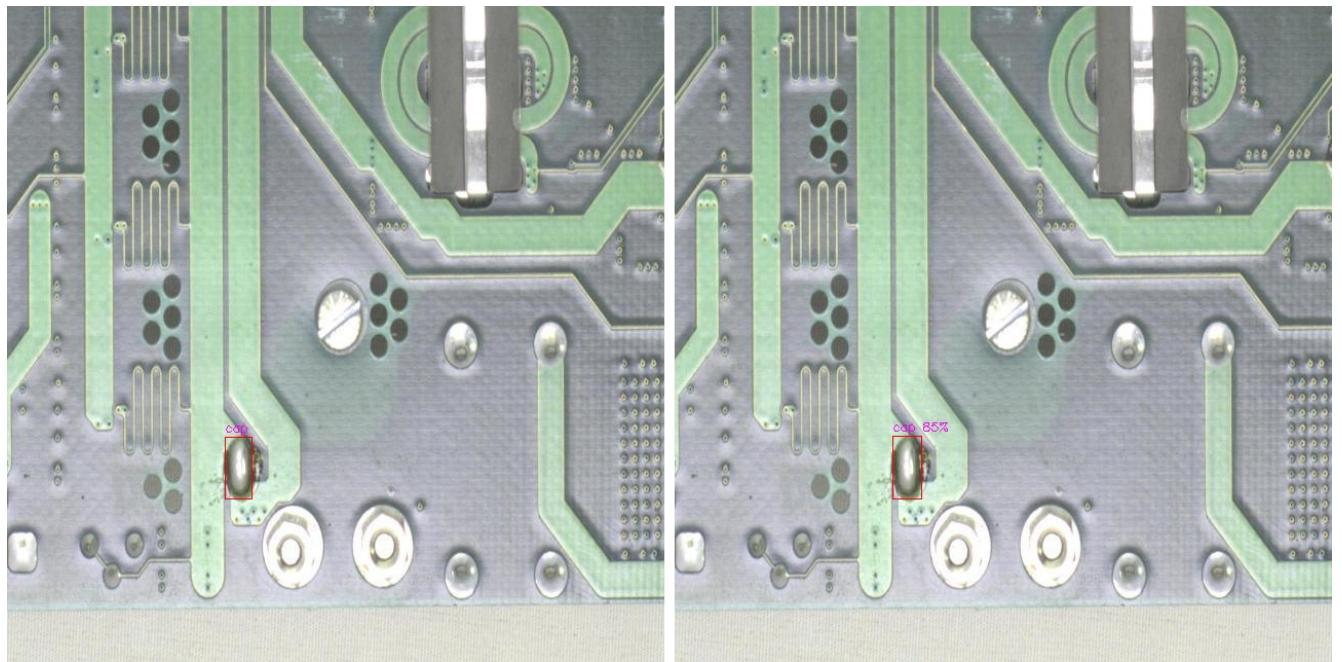


Figura 9: *Verdadero vs Predicho*

Como se puede ver en la figura 9 , el modelo tambien ha aprendido de forma errónea ya que se ve como un objeto es etiquetado como verdadero como si fuera un capacitor cuando realmente es una soldadura de estaño.

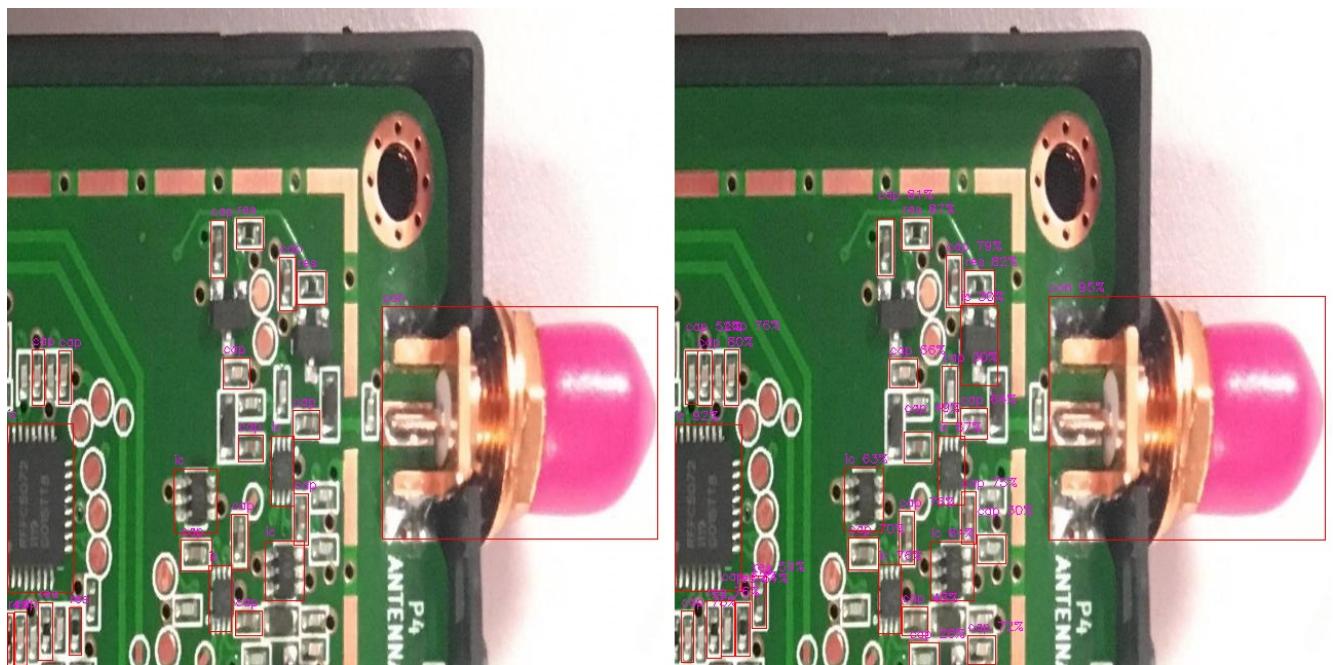


Figura 10:

También hay casos como el de la figura anterior en la que las etiquetas verdaderas no detectan todos los objetos que hay mientras que las predicciones si lo hacen pero se catalogan incorrectamente como errores. Esto se podría solucionar utilizando un dataset en el que las etiquetas sean mejor clasificadas.

4.2. Pruebas con imágenes ajenas al dataset

Se utilizaron algunos circuitos integrados para la prueba, que fueron una Raspberry PI 5 , esp32 y un regulador de voltaje.

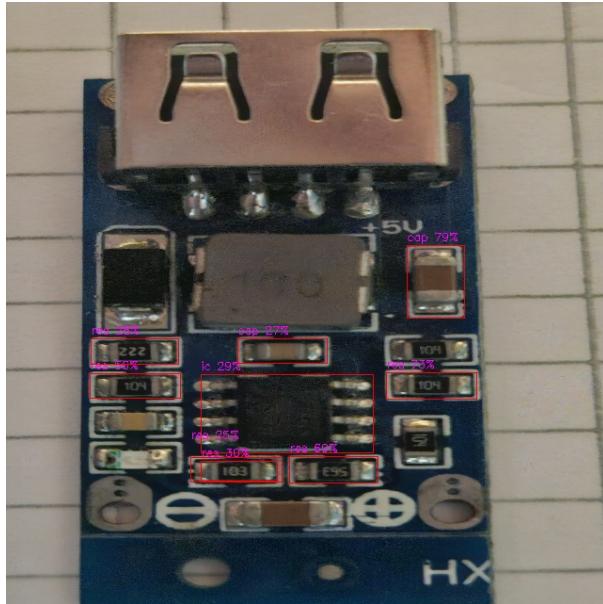


Figura 11: Regulador de voltaje

Vemos que el modelo le cuesta poder clasificar algunos objetos del pcb, probablemente se podría mejorar si imitamos el ángulo de captura que se utilizó para entrenar ya que esas imágenes son tomadas con instrumentos industriales de mejor calidad haciendo que la imagen no se distorsione por el ángulo de captura, ya que desde un principio este modelo tiene como fin ideal un uso industrial con otro tipo de cámaras.

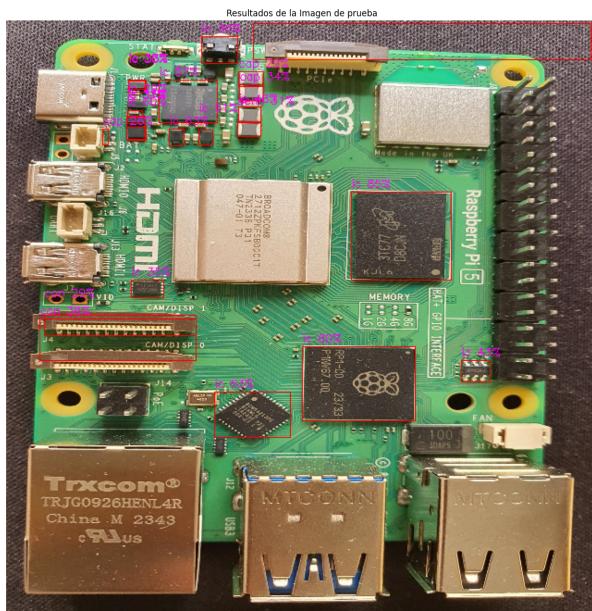


Figura 12: Raspberry PI 5

Otra opción podría ser entrenar el modelo con un dataset con más cantidad de imágenes y

utilizar más épocas.

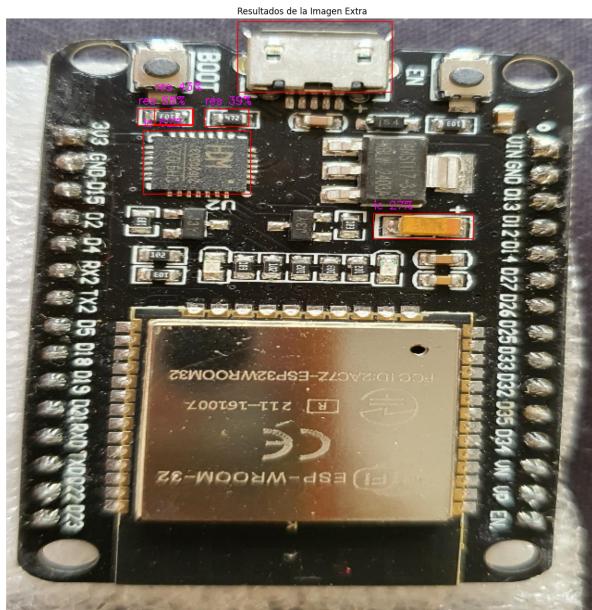


Figura 13: esp32

En el caso de la esp32 probablemente el problema fue el efecto ojo de pez en la fotografía en consecuencia los objetos se ven distorsionados.

Para finalizar, las pruebas no utilizaron una resolución nativa 640x640 por lo que debieron ser postprocesadas para que tengan esa resolución por lo que probablemente se distorsionó la imagen (el caso mas visible es el de la esp32) provocando menor rendimiento.

5. Conclusiones

Por lo que se ha podido ver a lo largo del informe vemos grandes características de las CNN, con muchas aristas para ser corregidas dandole un gran potencial, ideal para un manejo grande de datos; para industrias automatizadas como lo es la tecnológica, donde se pueden implementar modelos como este para ahorrar dinero y tiempo en verificar el producto terminado, obviamente hablamos de si el modelo obtiene una mejoría en la eficiencia.

Por otra parte YoloV8 tiene una gran mejoría en respecto a versiones anteriores (YoloV5) en eficiencia y tiempo, de utilizar las versiones más nuevas probablemente se notaría mucho más esos aspectos.

Con respecto a este modelo en particular es que si quisieramos aumentar la cantidad de componentes (aca se ha tenido en cuenta una cantidad pequeña) es bastante facil agregar más, el problema es encontrar un dataset correctamente etiquetado pero de ahí dependerá la inversión que quiera hacer la industria que quiera adoptarlo.

Ademas podemos adaptar estos modelos para lograr:

Detección Precisa de Detalles Pequeños: Los filtros pueden adaptarse para identificar defectos sutiles, como microfisuras o cortocircuitos.

Escalabilidad: Pueden manejar grandes volúmenes de imágenes de alta resolución, como las utilizadas en la inspección de PCBs.

Automatización y Eficiencia: Reducen la necesidad de inspección manual, acelerando los procesos de fabricación.

La capacidad de las CNN para aprender características espaciales complejas, junto con su eficiencia y adaptabilidad, las hace ideales para aplicaciones como la detección de defectos en PCBs. Su evolución a través de arquitecturas más profundas y eficientes, como AlexNet, VGG y ResNet, ha permitido resolver problemas que antes eran intratables con las ANN tradicionales.

Para concluir se debe destacar que con un mejor hardware podemos lograr una gran mejoría en tiempos y en eficiencia ya que podríamos agregar más épocas en un mismo lapso de tiempo, reduciendo el gran problema a la hora de querer realizar cambios a un modelo cuando se está en las primeras etapas de diseño, este en particular tardó aproximadamente 20 horas en entrenarse y cada vez que se quiso realizar cambios había que esperar una gran cantidad de tiempo en ver resultados.

6. Código

Código Completo GitHub