VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

BPH-YOLO: sistema de detección automatizada de incumplimientos de inocuidad alimentaria en tiempo real mediante visión por computadora en el Laboratorio de Alimentos de la Sede Janicho de la Universidad De Guanajuato

BPH-YOLO: automated detection system for food safety violations in real time through computer vision in the Food Laboratory at the Janicho Site, Universidad de Guanajuato

Maria de la Luz López González[1*], Daniela Lara Martínez[1], Jocelyn Soria Lesso[1]
[1]División de Ciencias de la Salud e Ingenierías, Universidad de Guanajuato, Campus Celaya-Salvatierra, Sede Janicho
*mdll.lopezgonzalez@ugto.mx

1 Resumen

Se construyó BPH-YOLO, un sistema de extremo a extremo —anotación, entrenamiento, validación y trackeo de personas mediante BoT-SORT con filtro de Kalman— capaz de detectar automáticamente 17 prácticas de cumplimiento e incumplimiento de Buenas Prácticas de Higiene (equipo de protección personal, higiene conductual y condiciones de superficie) en el Laboratorio de Alimentos de la Sede Janicho, Universidad de Guanajuato, mediante cinco arquitecturas YOLO (v8, v10, 11, 12 y 26) ajustadas por transferencia de aprendizaje desde COCO. Un diseño experimental de siete bloques aisló, con evidencia estadística, el efecto independiente del volumen de datos, la arquitectura, el tamaño del modelo y la estrategia frente al desbalance de clases (razón 18.08:1) sobre el desempeño del sistema. Los resultados confirman que el volumen de datos es el determinante principal del desempeño del sistema ($r = 0.61$ a 0.90, $p < .05$ en las cuatro arquitecturas evaluadas de forma completa), y que ninguna arquitectura domina de forma absoluta: YOLO11 es más robusta con datos escasos, mientras que YOLO26 alcanza el mejor desempeño global (mAP@0.5 = 0.54) con el conjunto completo. Ni ponderar la pérdida por clase ni especializar modelos mostraron una ventaja estadísticamente significativa sobre un único modelo de 17 clases, simplificando la recomendación de despliegue. Este trabajo se entrega como una validación metodológica rigurosa, con cada hallazgo respaldado estadísticamente y el diseño experimental documentado para su réplica o extensión; una de las cinco arquitecturas, YOLO12, quedó excluida de la comparación final, ya que en sesiones de entrenamiento prolongadas llegó a tardar casi 20 veces más de lo esperado por época y en inferencia resultó la más lenta del grupo pese a exigir un costo de cómputo teórico similar al de YOLO11, evidencia de que su mecanismo de atención es menos eficiente en el hardware disponible, tanto al entrenar como al detectar. Para el despliegue en cámara en vivo, el sistema ofrece tres modelos de detección de violaciones seleccionables según la prioridad de velocidad o precisión (YOLOv8-s, YOLO11-m y YOLO26-m), cada uno combinado con un YOLO11n preentrenado en COCO dedicado exclusivamente al trackeo de personas mediante BoT-SORT y filtro de Kalman, cuyos identificadores persistentes se asocian por Intersección sobre Unión con cada violación detectada. La latencia combinada de detección y trackeo se midió formalmente sobre 17,754 cuadros de video real de laboratorio (5 grabaciones, cámara fija, GPU NVIDIA RTX 5070): YOLOv8-s alcanzó el mejor rendimiento (26.44 FPS, ~37.8 ms/cuadro), seguido de YOLO11-m (19.15 FPS, ~52.2 ms/cuadro) y YOLO26-m (18.53 FPS, ~54.0 ms/cuadro); los tres modelos, junto con el YOLO11n dedicado al trackeo, se publicarán como repositorio público en GitHub.

Palabras clave: visión por computadora; arquitectura YOLO; inocuidad alimentaria; Buenas Prácticas de Higiene; FastAPI.

2 Introducción

La inocuidad alimentaria es un factor determinante para la salud pública mundial. Según estimaciones de la Organización Mundial de la Salud, las enfermedades transmitidas por alimentos afectan cada año a cerca de 600 millones de personas en el mundo y ocasionan más de 420,000 muertes, una carga sanitaria que recae de manera desproporcionada en el manejo deficiente de las prácticas de higiene durante la manipulación de

alimentos (OMS, 2022). Aunque los planes de Análisis de Peligros y Puntos Críticos de Control (HACCP) constituyen el estándar internacional para la gestión de la inocuidad, diversas revisiones señalan que el incumplimiento de los procedimientos prescritos y las conductas de higiene inadecuadas siguen siendo responsables de una proporción considerable de los brotes registrados. Esto ha desplazado el interés de la comunidad científica desde los controles exclusivamente documentales hacia enfoques conductuales y tecnológicos capaces de verificar, de forma objetiva y continua, si el personal cumple con el uso correcto del Equipo de Protección Personal (EPP), el lavado de manos y el manejo adecuado de superficies.

La inspección manual mediante auditorías periódicas, listas de verificación o supervisión directa presenta limitaciones estructurales bien documentadas: es intensiva en mano de obra, susceptible al sesgo del observador, difícil de sostener de manera continua y, por su naturaleza intermitente, propensa al efecto Hawthorne (la mejora temporal del comportamiento cuando el trabajador se sabe observado). En laboratorios universitarios de alimentos, esta limitación es aún más marcada: es materialmente imposible que un solo instructor observe a decenas de estudiantes durante toda la práctica, por lo que la visión por computadora mediante algoritmos de detección de objetos en tiempo real, ofrece una alternativa robusta para el monitoreo continuo y automatizado.

Frente a esa limitación estructural, la pregunta que orienta el diseño de BPH-YOLO es fácil de enunciar y compleja de resolver: ¿puede una cámara aprender a reconocer, con la misma seguridad que un inspector entrenado, el momento exacto en que una práctica de laboratorio incumple una norma de higiene? La respuesta podría encontrase en las redes neuronales convolucionales, arquitecturas capaces de aprender directamente de los píxeles los patrones que marcan la diferencia: desde bordes y texturas hasta siluetas. Así, logran distinguir una bata de laboratorio de un brazo descubierto, o una superficie limpia de una contaminada. Ese aprendizaje ocurre por retropropagación del error: la red compara su predicción con la anotación real y ajusta sus conexiones internas un poco hasta reducir el margen de error a un nivel aceptable. Reconocer un patrón, sin embargo, no es lo mismo que localizarlo en tiempo real dentro de un flujo de video continuo.

Ahí es donde entra la familia YOLO (You Only Look Once), que desde su formulación original resolvió la detección de objetos de un modo distinto al de sus predecesores: en vez de proponer regiones candidatas y clasificarlas una por una, procesa la imagen completa en una sola pasada, tratando la ubicación y la identidad de cada objeto como un mismo problema de regresión (Redmon et al., 2016). Esa decisión de diseño explica por qué la familia YOLO se ha mantenido como el estándar de facto para el monitoreo de cumplimiento en entornos regulados: ningún otro enfoque ofrece un balance comparable entre precisión y velocidad de inferencia cuando la exigencia es ver y decidir al mismo tiempo (Terven et al., 2023). Ese mismo balance tiene un costo: entrenar una red desde cero exige millones de ejemplos, cantidad limitada en un entorno de laboratorio universitario. La solución, es la adaptación de la transferencia de aprendizaje a partir de un modelo pre-entrenado que tiene la capacidad de distinguir formas y texturas sobre COCO (Common Objects in Context), un acervo de cerca de 330,000 imágenes en 80 categorías cotidianas, y ajustar solo lo necesario, el fine-tuning sobre el dominio específico de interés (Lin et al., 2014; Pan & Yang, 2010).

Sin embargo, un modelo que detecta guantes, cofias o derrames sirve de poco si esas detecciones no tienen una interpretación práctica. Ese significado lo aporta el marco normativo de la inocuidad alimentaria. Por un lado, la NOM-251-SSA1-2009 establece en México los requisitos mínimos de higiene para instalaciones, indumentaria y almacenamiento (DOF, 2010). Por otro, la ISO 22000:2018 integra estos principios con el sistema HACCP bajo un marco internacional, mientras que el Codex Alimentarius y la NMX-F-605-NORMEX complementan la base teórica. BPH-YOLO no solo cita esta normativa, sino que la utiliza directamente para estructurar su taxonomía de 17 clases y justificar técnicamente cada acción correctiva emitida.

Este recorrido — que va desde la percepción profunda por capas hasta la regulación normativa— no ocurre en el vacío. La detección automatizada de equipo de protección personal mediante arquitecturas YOLO es actualmente un área de investigación muy activa, cuyos avances permiten el alcance y las aportaciones de BPH-YOLO en un marco más amplio. Alashrafi et al. (2025) compararon YOLOv8n, YOLOv10n y YOLOv12n sobre un dataset de más de 31,000 imágenes anotadas en apenas 7 clases, y encontraron en YOLOv10n el mejor mAP@0.5 (85.7 %); su clase mejor detectada, la cofia, coincide con el patrón observado en el presente trabajo. Lo revelador es que, incluso con treinta veces más datos, el desbalance entre clases sigue siendo un problema latente. Esto ocurre porque las clases no aparecen con la misma frecuencia ni siguen una

distribución normal, sin importar el tamaño del dataset. En la auditoría real de un sistema de inocuidad, la frecuencia de cada clase es inherentemente desigual. Esta naturaleza no sintética complica la comparación entre categorías, a diferencia de los escenarios controlados donde sí es posible obtener métricas equiparables clase por clase.

El desbalance de clases es uno de los factores que condiciona con mayor consistencia el desempeño de los modelos de detección de objetos entrenados sobre datos del mundo real, donde la frecuencia de cada categoría rara vez es uniforme (He y Garcia, 2009). Esto es particularmente evidente en un sistema de auditoría de inocuidad alimentaria: ciertos incumplimientos, aunque críticos desde el punto de vista normativo, ocurren con menor frecuencia durante la operación cotidiana del laboratorio. Por ejemplo, un derrame o el uso de un celular son eventos naturalmente esporádicos, mientras que la presencia o ausencia de cofia ocurre en prácticamente cada interacción. En este sentido, el desbalance no es un defecto del muestreo que deba corregirse ajustando la recolección de datos, sino una propiedad inherente al fenómeno observado. Cuando ciertas clases están subrepresentadas, el proceso de optimización tiende a favorecer a las mayoritarias —ya que contribuyen más al gradiente promedio—, lo que puede generar una falsa sensación de alto desempeño si solo se reporta la métrica global. Esto oculta fallas precisamente en las clases raras que, pese a su baja recurrencia, conservan una alta relevancia para la inocuidad. Por ello, la literatura recomienda cuantificar explícitamente el desbalance antes de interpretar las métricas de desempeño, permitiendo atribuir las diferencias observadas a la disponibilidad de datos y no a limitaciones del modelo. Siguiendo este criterio metodológico, el desbalance del dataset de 17 clases se caracterizó mediante dos métricas complementarias: la razón de desbalance (Imbalance Ratio, IR) y la entropía de Shannon normalizada. En primer lugar, la IR representa la métrica estándar en la literatura para este propósito (He y Garcia, 2009):

Sabri et al. (2026), con YOLOv7-d6, utilizaron un dataset de tamaño comparable al de este estudio (1,500 imágenes, ampliadas a 2,859 mediante mosaico) y alcanzaron un mAP@0.5 de 0.918 Aunque esta cifra es superior a la reportada aquí, fue lograda con un número significativamente menor de clases y sin enfrentar el desbalance extremo (831:9) que aborda BPH-YOLO. La lectura no es que el volumen de datos sea la única variable que importa, sino que la amplitud de la taxonomía y el desbalance de clases pesan al menos tanto como el tamaño del dataset. Por ello resulta pertinente analizar cómo el número de clases identificadas puede introducir diferencias significativas entre estudios que comparan distintas generaciones de arquitecturas YOLO, considerando que un menor número de clases no exime al modelo del efecto del desbalance sobre su desempeño. La taxonomía de clases es, en sí misma, un objeto evaluable; sin embargo, el número de clases necesario para validar un sistema de inocuidad alimentaria no debería reducirse al equipo de protección personal, aunque en la literatura sea este el que concentra mayor atención durante el entrenamiento de arquitecturas YOLO. Li et al. (2025), con OAM-YOLO, atacaron el mismo problema desde la arquitectura: combinaron atención agregada, convolución adaptativa y pérdidas especializadas para mitigar el desbalance de clases en objetos pequeños, exactamente el tipo de objetos, como joyería o celulares, que BPH-YOLO también debe detectar. Aun con esas mejoras, concluyeron que el desbalance y el ruido visual siguen exigiendo trabajo adicional.

El estudio de Çarklı Yavuz (2026) ofrece la comparación más cercana al diseño experimental de este trabajo, al evaluar 30 configuraciones (cinco escalas de modelo y dos arquitecturas: YOLO11 y YOLO26) sobre tres datasets, uno de los cuales comparte las 17 clases utilizadas aquí bajo condiciones controladas idénticas. Su hallazgo principal es doble. Primero, demuestra que YOLO11 supera a YOLO26 en escalas pequeñas, pero pierde ventaja en las grandes; esta observación orienta la selección de arquitecturas y tamaños a comparar, aspecto que se evalúa formalmente en el bloque experimental A4 de este proyecto. Segundo, revela que entrenar en escalas pequeñas no implica necesariamente una pérdida de eficiencia frente a modelos más grandes; sin embargo, esta ventaja varía entre arquitecturas. Esto sugiere que la relación entre la escala del modelo y su desempeño debe verificarse de forma específica y no asumirse como una regla general para toda la familia YOLO. En conjunto, estos antecedentes transmiten un mensaje claro: el desbalance de clases en la detección de EPP continúa siendo un reto sin resolver —incluso en datasets uno o dos órdenes de magnitud mayores—, y casi ningún trabajo previo extiende su taxonomía más allá del equipo de protección personal hacia comportamientos de higiene o condiciones del entorno.

Partiendo de las premisas de la arquitectura YOLO, BPH-YOLO realiza tres aportaciones concretas. La primera es su taxonomía de 17 clases, la cual integra equipo de protección personal, higiene conductual y condiciones de superficie, superando la limitación hacia el EPP convencional. La segunda aportación es

metodológica y representa el aporte central del trabajo: incorpora salvaguardas para datasets pequeños desarrollados en entornos educativos, como la partición por video de origen para prevenir la fuga de datos, la paginación determinista en el conteo de anotaciones y la selección del modelo por la época de mejor fitness real —detalles que, de omitirse, habrían comprometido la validez de las métricas. La tercera es prospectiva: el seguimiento de personas para retroalimentación formativa individualizada se concibe como una línea de investigación a futuro, manteniendo la congruencia con el alcance de este estudio, enfocado en el diseño, validación y hallazgos del sistema. Trazado este panorama, a continuación, se detalla la metodología empleada.

3 Metodología

3.1 Implementación del sistema BPH-Yolo

BPH-YOLO implementa un ciclo de aprendizaje supervisado en cuatro etapas: captura y anotación de imágenes y fotogramas de video mediante cajas delimitadoras en una interfaz web con control de versiones; curación del conjunto de datos, sincronizando las anotaciones activas al formato YOLO con una partición entrenamiento/validación diseñada para evitar fuga de información; entrenamiento por transferencia desde pesos preentrenados en COCO (Lin et al., 2014), ajustados sobre las 17 clases del dominio y comparando cinco generaciones de arquitectura (YOLOv8, YOLOv10, YOLO11, YOLO12 y YOLO26); y validación y despliegue, donde el modelo de mejor ajuste se activa para inferencia en vivo o sobre video grabado. Estas etapas se sostienen en una arquitectura cliente-servidor de cuatro capas: una interfaz web (panel de control, visor en vivo, anotación y reportes), un servidor que atiende tanto la inferencia en tiempo real como las operaciones de carga, anotación y entrenamiento, un motor de visión que evalúa cada fotograma contra las 17 clases normativas, y un almacenamiento dual (local o en la nube). A diferencia de un sistema de visión convencional, BPH-YOLO vincula cada detección con su justificación normativa y muestra en tiempo real la acción correctiva correspondiente, exportable como reporte de auditoría.

3.2 Esquema de clases y alineación normativa

Las 17 clases de detección se organizaron en cuatro estaciones operativas, alineadas con la NOM-251-SSA1-2009, la NMX-F-605-NORMEX y la ISO 22000 (Tabla 1): Equipo de Protección Personal (EPP), Higiene Personal, Superficies, y Eventos Críticos de contaminación cruzada. Cada clase tiene asociada una justificación normativa y una acción correctiva automatizada.

Tabla 1 Mapeo normativo de las clases de detección del Sistema BPH-YOLO.

Estación	Clases detectadas	Alineación normativa	Justificación / acción correctiva
Equipo de protección personal (EPP)	Bata, cofia, cubrebocas, guantes (y sus ausencias).	NOM-251 § 5.14.4; NMX-F-605 § 5.2; ISO 22000 cl. 8.2.	Barrera física contra la contaminación directa del alimento (fibras, cabello, gotas de saliva, contacto manual). Acción correctiva: colocación inmediata del equipo de protección personal faltante antes de continuar la actividad.
Higiene Personal	Celular, joyería, comer/beber, lavado de manos, cabello expuesto.	NOM-251 § 5.14.1, § 5.14.3; NMX-F-605 § 5.2; ISO 22000 cl. 8.2.	Reduce el riesgo de contaminación cruzada por contacto con manos, objetos personales o superficies contaminadas. Acción correctiva: retirar el objeto de riesgo y realizar lavado de manos antes de continuar.
Superficies	Área limpia / contaminada, derrames.	NOM-251 § 5.1; NMX-F-605 § 5.1; ISO 22000 cl. 8.2.4.	Previene el crecimiento de microorganismos por residuos de alimento o suciedad en superficies de contacto. Acción correctiva: limpieza y desinfección inmediata de la superficie o el derrame detectado.
Eventos Críticos	Contaminación cruzada.	NOM-251 § 5.3.2; NMX-F-605 § 5.6; ISO 22000 cl. 8.5.2.	Evita que microorganismos se transfieran entre alimentos, superficies o utensilios de distintas áreas o etapas del proceso. Acción correctiva: detener la actividad, desinfectar el punto de contacto y separar los productos involucrados.

3.3 Conjunto de datos experimentales

El acervo recabado para este proyecto asciende a 1,643 imágenes, con un total de 10,254 cajas delimitadoras anotadas. De este conjunto se fijó un examen de 154 imágenes como conjunto de prueba independiente; las 1,489 restantes conformaron el pool de entrenamiento. La Tabla 2 muestra la distribución de cajas por clase.

Tabla 2. Distribución de cajas delimitadoras por clase, base para el cálculo de la razón de desbalance (IR) y la entropía de Shannon normalizada

Tipo	Clase (cajas delimitadoras)
Incumplimiento	Superficie contaminada (2,043); Sin guantes (1,356); Sin cubrebocas (882); Sin bata de laboratorio (832); Sin cofia / cabello descubierto (806); Joyería / accesorios (725); Contaminación cruzada (387); Cabello expuesto (261); Derrame de sustancia (148); Comer o beber (118); Uso de celular (113). **Total: 7,671**
Cumplimiento	Bata de laboratorio (775); Superficie limpia (725); Cofia / redecilla (508); Lavado de manos (237); Cubrebocas / mascarilla (182); Guantes de protección (156). **Total: 2,583**

3.4 Formulación matemática: Función de pérdida y métricas de evaluación

El optimizador minimiza, para cada predicción positiva, una combinación ponderada de tres funciones de pérdida, registradas en el sistema como $box_{\text{loss}}, cls_{\text{loos}}$ y dfl_{loss}. La pérdida de localización se calcula mediante CIoU (Zheng et al., 2020), que extiende la Intersección sobre Unión penalizando adicionalmente la distancia entre los centros de la caja predicha y la de referencia, así como la disparidad en su relación de aspecto. La pérdida de distribución focal (DFL) (Li et al., 2020) predice, para cada borde de la caja delimitadora, una distribución discreta de probabilidad sobre bins enteros vecinos, en lugar de regresar un valor escalar continuo. La pérdida de clasificación se calcula mediante entropía cruzada binaria sobre las 17 clases del sistema. Estos tres términos se combinan en la función de pérdida total, $L_{total} = \lambda_{box} \cdot L_{CIoU} + \lambda_{cls} \cdot L_{cls} + \lambda_{dfl} \cdot L_{DFL}$, ponderados con las ganancias por defecto de Ultralytics ($\lambda_{box} = 7.5, \lambda_{cls} = 0.5, \lambda_{dfl} = 1.5$), sin modificaciones en este proyecto.

Para evaluar el grado de ajuste del modelo sobre datos no vistos durante el entrenamiento, se define la Intersección sobre Unión (IoU) entre una caja predicha B_p y una anotación de referencia B_g, definida por $IoU(B_p, B_g) = \frac{|B_p \cap B_g|}{|B_p \cup B_g|}$. Una predicción se clasifica como verdadero positivo (VP) si $IoU \geq 0.5$,convención introducida por Everingham et al. (2010), y coincide en clase con la anotación correspondiente; en caso contrario, se registra como falso positivo (FP). Toda anotación de referencia sin una predicción correspondiente se registra como falso negativo (FN). A partir de estos tres conteos se derivan la precisión (P), el recall (R) y su combinación en la métrica F1:

$$P = \frac{VP}{VP + FP}, \qquad R = \frac{VP}{VP + FN}, \qquad F_1 = \frac{2PR}{P + R} \tag{1}$$

La Precisión Promedio (Average Precision, AP) se define como el área bajo la curva precisión-recall, interpolada en 101 puntos siguiendo el protocolo de evaluación de COCO (Lin et al., 2014), más estricto que el método original de 11 puntos empleado en PASCAL VOC (Everingham et al., 2010). El mAP@0.5 promedia el AP de todas las clases evaluables bajo un umbral fijo de $IoU \geq 0.5$, mientras que el mAP@0.5:0.95 (la métrica completa de COCO, y el criterio que Ultralytics utiliza para seleccionar el mejor punto de entrenamiento (fitness)) promedia adicionalmente sobre diez umbrales de IoU, de 0.50 a 0.95 en incrementos de 0.05: $mAP@[0.5{:}\,0.95] = \frac{1}{10}\sum_{t\in\{0.50,0.55,\ldots,0.95\}} AP_{t}.$, penalizando con mayor severidad la imprecisión en la localización de la caja delimitadora que el criterio único de mAP@0.5.

3.5 Cuantificación de desbalance de clases

La razón de desbalance (*Imbalance Ratio*, IR) es la métrica estándar en la literatura de aprendizaje con clases desbalanceadas para expresar la magnitud del desbalance entre la clase más y la menos representada (He & Garcia, 2009). Se define como: $IR = \frac{n_{max}}{n_{min}}$ donde $n_{máx}$ y $n_{mín}$ son los conteos de la clase más y la menos frecuente, espectivamente. Con los valores del dataset, IR = 2,043 / 113 = 18.08, lo que significa que por cada caja delimitadora de “uso del celular” (la clase menos representada) existen aproximadamente 18 de “superficie contaminada” (la clase más representada). El IR tiene, sin embargo, una limitación relevante: al

construirse únicamente a partir de las dos clases extremas, ignora por completo la distribución de las 15 clases intermedias. Por ello se complementó con la entropía de Shannon normalizada, que sí incorpora la distribución completa de las 17 clases:

$$H = -\sum_{i=1}^{K} p_i \ln(p_i), \quad H_{norm} = \frac{H}{\ln K} \in [0,1] \tag{6}$$

donde $p_i = n_i/N$ es la proporción de instancias de la clase i, y $k = 17$ es el número total de clases. Un valor de $H_{norm} = 1$ corresponde a un balance perfecto (todas las clases con igual frecuencia), mientras que valores cercanos a 0 indican una concentración fuerte en pocas clases. El dataset ampliado obtuvo $H_{norm} = 0.891$, un valor relativamente cercano al balance perfecto pese al IR de 18.08. Ambas cifras no son contradictorias, sino complementarias: indican que la mayoría de las 17 clases se distribuyen en un rango intermedio razonablemente equilibrado, y que el desbalance extremo capturado por el IR se concentra solo en un par de clases verdaderamente raras (uso_celular, comer_beber y derrame_sustancia), sin que estas dominen la lectura global de la distribución.

3.6 Diseño experimental y análisis estadístico

A partir de esta caracterización del desbalance se diseñó un conjunto de siete experimentos, con el fin de determinar con evidencia estadística si el techo de desempeño responde principalmente al volumen de datos o al desbalance entre clases, y qué estrategia priorizar en consecuencia.

Línea base por clase (Experimento A1) estableció el desempeño de referencia de las 17 clases bajo la configuración estándar (precisión, recall, F1, AP50, AP50-95), comparando el promedio macro contra el ponderado por soporte, que aísla el efecto del desbalance; y calculando la correlación de Pearson entre ln(instancias de entrenamiento) y AP50 por clase, con su prueba t de significancia.

Curva de aprendizaje (Experimento A2) determinó si el desempeño estaba limitado por la cantidad de datos, entrenando el mismo modelo con fracciones anidadas y crecientes (30/50/70/100%) contra el mismo examen fijo, ajustando una regresión log-lineal $\text{mAP50} = a + b \cdot ln(n_{train})$(con su r^2) y calculando la ganancia marginal por cada 100 imágenes nuevas.

Ablación de ponderación por frecuencia inversa de clase (cls_{pw}, experimento A3) puso a prueba si ponderar la pérdida por frecuencia inversa de clase mitigaba el desbalance, comparando tres valores (0.0, 0.5, 1.0) bajo condiciones idénticas en las cinco arquitecturas, con una prueba de Wilcoxon sobre las cinco diferencias de mAP50 ($cls_{pw} = 0.5$ vs. 0.0) para distinguir un efecto sistemático de uno específico por arquitectura.

Comparación de arquitecturas (Experimento A4) exploró qué generación (YOLOv8, YOLOv10, YOLO11, YOLO12, YOLO26) y tamaño (n/s/m) ofrecía el mejor equilibrio desempeño-costo, entrenando todas las combinaciones sobre el mismo dataset y examen fijo. Al entrenarse una sola vez por celda con una semilla fija, el análisis fue descriptivo, un mapa de mAP50-95 por arquitectura/tamaño y la correlación entre número de parámetros y desempeño dentro de cada arquitectura, ya que aislar la varianza de la inicialización aleatoria de la propia de la arquitectura habría requerido múltiples semillas por configuración, fuera del alcance de cómputo de esta estancia.

Solo clases de lata representación (Experimento B1) y baja representación (Experimento B2) exploraron si conviene especializar modelos por grupo de clases en vez de mantener uno generalista: B1 entrenó solo con las clases de alta representación (techo de desempeño alcanzable como mejor caso), comparándolo contra esas mismas clases dentro del sistema completo (A1); B2 replicó el ejercicio con las clases de baja representación, comparándolo contra su desempeño dentro del sistema completo ponderado (A1/A3).

Comparación entre especializar o ponderar (Experimento C) sintetizó esta evidencia sin entrenamiento adicional: contrastó el desempeño de las clases raras en A3 (ponderación dentro del sistema completo) contra

B2 (modelo especializado), con una prueba de Wilcoxon pareada por clase sobre las diferencias A3−B2, para determinar si especializar ayuda de forma sistemática o si basta un único modelo de 17 clases bien ponderado. La Tabla 3 resume arquitecturas, tamaños y fracciones evaluados en cada experimento.

Tabla 3. Diseño de experimentos para la validación de las arquitecturas YOLO en BPH-YOLO

Clave	Arquitectura(s) / Tamaño(s) / Fracción	Métricas
A1**	YOLO11-s, 100%	mAP50, mAP50-95, Precisión, Recall, F1 (por cada una de las 17 clases más promedios globales)
A2**	4 arquitecturas completas (YOLO11, YOLOv10, YOLOv8, YOLO26) × (n/s/m) × (30/50/70/100%). YOLO12 fue excluido de la comparación final por costo computacional desproporcionado*.	mAP50, mAP50-95, Precisión, Recall, F1 por combinación
A3**	5 arquitecturas × s × 100%, cls_pw ∈ {0.0, 0.5, 1.0}	mAP50, mAP50-95 por (arquitectura, cls_pw)
A4**	5 arquitecturas × n/s/m (l pendiente) × 100%	mAP50, mAP50-95, F1, parámetros, tiempo por época
B1**	YOLO11 y YOLO26 × n/s/m, 100%, subconjunto de 9 clases de alta representación (6 corridas)	mAP50, mAP50-95, Precisión, Recall, F1 (global y por clase incluida)
B2**	YOLO11 y YOLO26 × n/s/m, 100%, subconjunto de 8 clases de baja representación (6 corridas)	Igual que B1, para el conjunto de clases raras
C**	Síntesis de A3, B1 y B2 (sin entrenamiento adicional)	AP50 por clase (alta y baja representación): sistema completo con cls_pw=0.5 (A3) contra sistema especializado (B1/B2), para YOLO11 y YOLO26, prueba de Wilcoxon pareada por clase (4 comparaciones: 2 arquitecturas × 2 grupos de clase)

***Nota sobre la exclusión de YOLO12 (aplicable a los experimentos A2/A4).** En corrida aislada (A3, tamaño "s") YOLO12 entrenó a ritmo normal (0.98 min/época, la segunda más rápida de las cinco arquitecturas); sin embargo, en la sesión extensa de A2/A4 (tamaño "m", múltiples arquitecturas entrenando de forma consecutiva por más de 40 horas) su ritmo se degradó a ~19 min/época , un factor de ~19 veces sobre su propio ritmo normal, confirmado en dos corridas independientes; con la GPU al 100% de utilización y la VRAM casi saturada (11.66/12.23 GB, ~95%). Esta lentitud es consistente con su latencia de inferencia: pese a tener GFLOPs prácticamente idénticos a YOLO11 (48.5 vs. 48.6), YOLO12 resultó 2.4 veces más lento (106.2 ms vs. 75.9 ms), lo que indica que su costo no proviene del volumen de cómputo teórico, sino de una eficiencia real menor de sus operaciones de atención en este hardware , consistente en toda condición de inferencia, pero manifiesta en entrenamiento solo bajo sesiones prolongadas, posiblemente por fragmentación de memoria acumulada tras decenas de entrenamientos consecutivos en el mismo proceso. Por este costo computacional desproporcionado, YOLO12 se excluyó de la comparación final de A2/A4. ****Nota sobre configuración de entrenamientos:** Todos los entrenamientos compartieron la siguiente configuración: resolución de entrada (imgsz) de 960 px; 50 épocas máximas con early stopping a las 15 épocas sin mejora; batch size de 8; tasa de aprendizaje inicial lr0=0.01 con un factor de decaimiento final lrf=0.01 (tasa final efectiva ≈0.0001), 3 épocas de calentamiento; semilla fija (seed=42); y el siguiente esquema de aumentación: hsv_h=0.015, hsv_s=0.5, hsv_v=0.4, degrees=5.0, translate=0.1, scale=0.25, fliplr=0.5, mosaic=0.5, mixup=0.0. Los pesos iniciales se preentrenaron sobre COCO-80 (Ultralytics). El examen fijo correspondió al 9.4% del dataset (154/1,643 imágenes), congelado una sola vez y compartido por todos los experimentos (A2, A3, A4, B1 y B2).

Como complemento al experimento A4, y dado que el objetivo final del sistema es operar como una herramienta de auditoría reactiva en tiempo real, se incorporó un octavo experimento de *benchmarking* computacional: mientras A1–C caracterizan qué tan bien detecta el modelo, este bloque evalúa qué tan rápido lo hace. El análisis de rendimiento se realizó midiendo dos métricas fundamentales sobre el flujo de inferencia soportado por FastAPI y WebSockets. La latencia total de inferencia, expresada en milisegundos, se descompone en tres fases: pre-procesamiento, propagación hacia adelante en la red (inferencia) y post-procesamiento: $t_{total} = t_{pre} + t_{inf} + t_{post}$. .Los fotogramas por segundo (FPS), que definen la fluidez de la transmisión de video en tiempo real, se derivan directamente de la latencia total,

$$FPS = \frac{1000}{t_{pre}+t_{inf}+t_{post}}.$$

Este benchmarking se ejecutó para las cinco arquitecturas comparadas en A4 (YOLOv8, YOLOv10, YOLO11, YOLO12 y YOLO26), contrastando su latencia y FPS contra el mismo dataset y examen fijo, con el fin de determinar empíricamente qué configuración mantiene un procesamiento en tiempo real sin sacrificar de forma significativa las métricas de precisión (mAP@0.5 y mAP@0.5:0.95) reportadas en los bloques anteriores. El criterio de aceptación operativa se fijó en una latencia inferior a 100 ms por fotograma sobre hardware de gama media (RTX 5070), meta de rendimiento establecida para el despliegue del sistema en el laboratorio.

3.7 Infraestructura de cómputo y entorno de desarrollo

El entrenamiento, la validación y la evaluación de inferencia se llevaron a cabo en un entorno controlado, sobre Python 3.10 y PyTorch como marco de trabajo de aprendizaje profundo. La infraestructura de cómputo consistió en una estación de trabajo con procesador Intel Core i9-12900KF (12.ª generación), 64 GB de memoria RAM y placa base ASUS PRIME B760M-A D4, operando bajo Windows 11 Pro con seguridad basada en virtualización (VBS) e integridad de código aplicada por hipervisor; el procesamiento se aceleró mediante una GPU NVIDIA GeForce RTX 5070. El motor de visión computacional se implementó sobre Ultralytics (≥8.3.0) para el entrenamiento y la inferencia YOLO, OpenCV (≥4.10.0) para el procesamiento de fotogramas y lap (≥0.5.12) para la asociación de detecciones, con scikit-learn (≥1.5.0), SciPy (≥1.14.0), pandas (≥2.2.0), matplotlib (≥3.9.0) y seaborn (≥0.13.0) para el análisis estadístico y la generación de figuras reportados en este artículo. El backend del sistema se construyó sobre FastAPI (≥0.115.0) y Uvicorn (≥0.32.0) para la comunicación asíncrona vía WebSockets y REST, con Jinja2 (≥3.1.4) para el renderizado de plantillas y Supabase (≥2.0.0) para el almacenamiento en la nube.

La anotación de las cajas delimitadoras se realizó mediante una herramienta web desarrollada específicamente para este proyecto, con un lienzo interactivo (canvas HTML5) que permite trazar una caja delimitadora por objeto y asignarle una única clase de entre las 17 definidas en la taxonomía. Cada caja se almacena en el formato normalizado estándar de la familia YOLO, la tupla (c, x_c, y_c, w, h), con coordenadas de centro, ancho y alto expresadas como fracción de las dimensiones de la imagen—, lo que garantiza que las anotaciones sean independientes de la resolución original de captura. El proceso de anotación estuvo a cargo del equipo de investigación, y la concordancia entre anotadores se evaluó mediante el coeficiente Kappa de Cohen, con un umbral mínimo aceptado de $\kappa \geq 0.75$ como criterio de control de calidad. Para preservar la trazabilidad del proceso, cada reedición de una imagen ya anotada se gestionó mediante versionado por soft-delete: las anotaciones previas se marcan como obsoletas únicamente después de confirmar que las nuevas se insertaron correctamente, evitando que un fallo de escritura deje una imagen sin anotaciones activas.

El desarrollo de este sistema se realizó sobre esta misma estación, mediante un flujo de trabajo asistido por agentes de inteligencia artificial integrados en el entorno de desarrollo (IDE). El desarrollo general de la plataforma se llevó a cabo de la mano de Claude Code (Anthropic, 2026) y GitHub Copilot, ambos integrados como extensiones dentro de Visual Studio Code: el primero para planificar, ejecutar y verificar tareas completas (leer archivos, escribir funciones, correr pruebas), y el segundo principalmente para autocompletado de código y sugerencias en línea durante la edición. De manera complementaria, se utilizó Google Antigravity, un IDE independiente construido sobre una bifurcación de Visual Studio Code (Google, 2025), específicamente para diseñar una herramienta independiente de validación local, que permite ejecutar y comparar en distintos equipos de cómputo los modelos entrenados con mejores métricas, previo a su integración a la plataforma principal. En todos los casos, el equipo de investigación conservó la revisión y decisión final sobre cada cambio incorporado al repositorio, siguiendo la práctica recomendada de supervisión activa en el desarrollo de software asistido por IA (Willison, 2025).

4 Resultados y discusión

De los ocho experimentos planeados (A1-C, más el benchmarking de latencia), siete concluyeron en su totalidad. El experimento A4, la comparación de arquitecturas en su configuración más exigente quedó parcial; la razón, documentada más adelante, constituye uno de los hallazgos del estudio.

4.1 Cuantificación el desempeño base por clase (A1)

El primer experimento estableció el desempeño de referencia del sistema sin ninguna estrategia adicional frente al desbalance de clases. El mAP@0.5 macro, promedio simple entre las 17 clases, fue de 0.361, frente a 0.463 al ponderar por soporte (número de ejemplos por clase). La diferencia de 0.102 puntos cuantifica directamente el efecto del desbalance sobre el desempeño agregado. A nivel de clase individual, el rango fue amplio: ***sin bata de laboratorio*** (393 instancias de entrenamiento) alcanzó un AP50 de 0.738, mientras que ***derrame de sustancia*** (89 instancias) se ubicó en 0.005. La correlación entre volumen de datos por clase y desempeño fue estadísticamente significativa ($r = 0.592$, $p = .016$), confirmando que el volumen de datos es

un predictor relevante del desempeño por clase, hallazgo que orientó el diseño del resto del diseño experimental.

4.2 Medición del efecto del volumen de datos (A2)

Se entrenó el mismo modelo con fracciones crecientes del conjunto de datos (30/50/70/100%), evaluando siempre contra el mismo examen fijo. El mAP50 aumentó de 0.401 a 0.478, con incrementos decrecientes por cada fracción adicional (+0.029, +0.016, +0.032), consistente con un patrón de rendimientos decrecientes. Este comportamiento se replicó en las cuatro arquitecturas evaluadas de forma completa (Figura 1), con correlaciones significativas entre volumen de datos y desempeño en todas ellas (r = 0.61 a 0.90). Estos resultados confirman que el sistema se encontraba limitado principalmente por el volumen de datos disponible, no por factores de configuración.

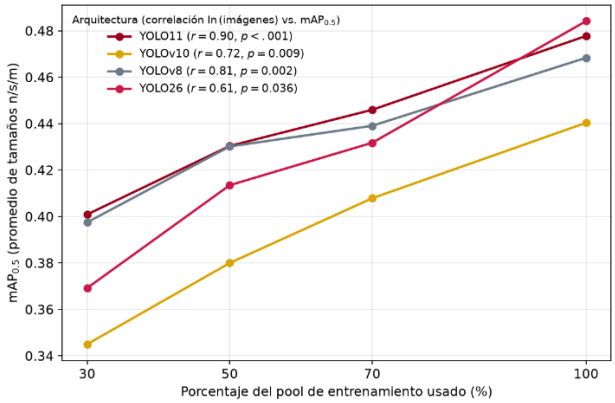


Figura 1. *Curva de aprendizaje por arquitectura: mAP@0.5 (promedio de tamaños n/s/m) en función del porcentaje del pool de entrenamiento utilizado (30/50/70/100%), evaluado siempre contra el mismo examen fijo de 154 imágenes. Entre paréntesis, el coeficiente de correlación de Pearson (r) entre ln(imágenes de entrenamiento) y mAP50, y su significancia estadística, calculados sobre las 12 combinaciones tamaño×fracción de cada arquitectura. Todas las arquitecturas muestran una relación positiva y significativa entre volumen de datos y desempeño, confirmando que el sistema está limitado por datos; YOLO11 exhibe la relación más fuerte y predecible (r=0.90), mientras que YOLO26 pese a tener la correlación más débil (r=0.61) alcanza el mejor desempeño absoluto al 100% de los datos, evidenciando una curva de aprendizaje más irregular pero de mayor límite.*

4.3 Prueba de la ablación por frecuencia inversa de clase (A3)

Se evaluó si ponderar la pérdida de clasificación por frecuencia inversa de clase mitigaba el efecto del desbalance, comparando tres niveles de ponderación en las cinco arquitecturas (15 corridas, Figura 2). El mAP50 promedio disminuyó levemente al aumentar la ponderación, sin significancia estadística (Friedman, p = .449). Este resultado nulo se limita al desempeño agregado, dominado por las clases de mayor representación; no permite inferir el efecto de la ponderación sobre las clases raras específicamente, pregunta que se aborda en los experimentos siguientes.

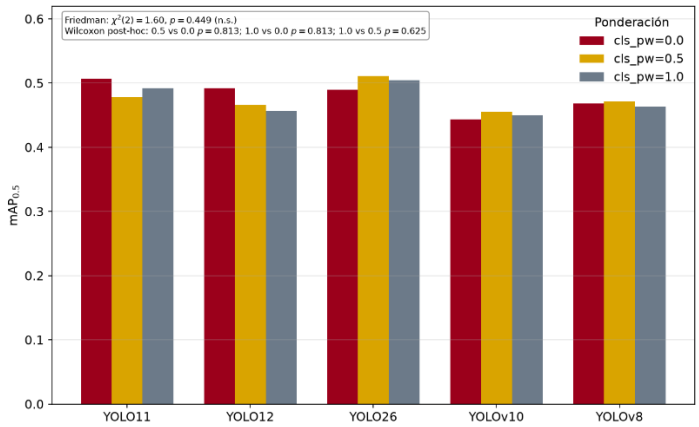


Figura 2. Efecto del parámetro cls_{pw} (ponderación de la pérdida de clasificación por frecuencia inversa de clase) sobre el mAP@0.5, para las cinco arquitecturas evaluadas en el Experimento A3 (tamaño 's', 100% del pool de entrenamiento). La prueba de Friedman para medidas repetidas (n=5 arquitecturas, 3 condiciones) no encontró un efecto estadísticamente significativo de cls_{pw} sobre el mAP@0.5 agregado ($\chi^2(2)=1.60$, $p=.449$), confirmado por las tres comparaciones post-hoc de Wilcoxon entre pares de condiciones ($p \geq .625$ en todos los casos). La ausencia de efecto a nivel agregado no implica ausencia de efecto sobre las clases raras específicamente.

4.4 La Comparación de arquitecturas (A4)

Comparación entre las cuatro arquitecturas completas no arrojó una arquitectura dominante, sino dos perfiles de desempeño diferenciados según la disponibilidad de datos:

Tabla 4. Comparación de mAP@0.5 entre YOLO11 y YOLO26 según la disponibilidad de datos de entrenamiento

Arquitectura	Fracción reducida (30%, modelo s)	Fracción completa (100%, modelo l)
YOLO11	0.3805 (aquí gana)	0.4890
YOLO26	0.2869	0.5425

En condiciones de datos limitados, YOLO11 obtuvo el mejor desempeño y la relación más fuerte y predecible entre volumen de datos y resultado (r = 0.90, la más alta del estudio). YOLO26, en cambio, obtuvo el desempeño más bajo del conjunto en esa misma condición (0.2869, el valor mínimo entre las 48 combinaciones evaluadas), pero alcanzó el mejor resultado absoluto del estudio (0.5425) al disponer del dataset completo y el tamaño de modelo mayor. Este comportamiento es consistente con lo reportado por Çarklı Yavuz (2026): la arquitectura más reciente no domina de forma automática en todas las condiciones de escala. La elección de arquitectura, en consecuencia, depende del volumen de datos disponible en el momento del entrenamiento.

4.5 Especialización por representatividad de clase (B1/B2)

Se evaluó si entrenar modelos independientes por grupo de clases mejoraba el desempeño frente al sistema generalista. Con las nueve clases de mayor representación (B1), el desempeño fue alto y estable (0.55–0.60) independientemente del tamaño de modelo. Con las ocho clases de menor representación (B2), el desempeño disminuyó, con tendencias opuestas por arquitectura: en YOLO11, el desempeño se redujo al aumentar el tamaño del modelo (posible sobreajuste dado el conjunto de entrenamiento reducido); en YOLO26, aumentó. Este patrón es consistente con el observado en A4.

4.6 Contrastando especializar frente a ponderar (C)

Se comparó el desempeño de las clases raras entre el sistema completo con ponderación (A3) y el modelo especializado (B2). No se encontró diferencia estadísticamente significativa en ninguna de las dos

arquitecturas (Wilcoxon, p = .742 y p = .250). El desglose por clase (Figura 3) mostró un patrón heterogéneo: ***guantes*** **y** ***derrame*** se deterioraron notablemente al especializar, mientras que ***comer o beber*** mejoró, sin una dirección consistente entre clases o arquitecturas. Esta evidencia respalda mantener un único modelo de 17 clases con ponderación por defecto, en lugar de desplegar modelos especializados por grupo de clases.

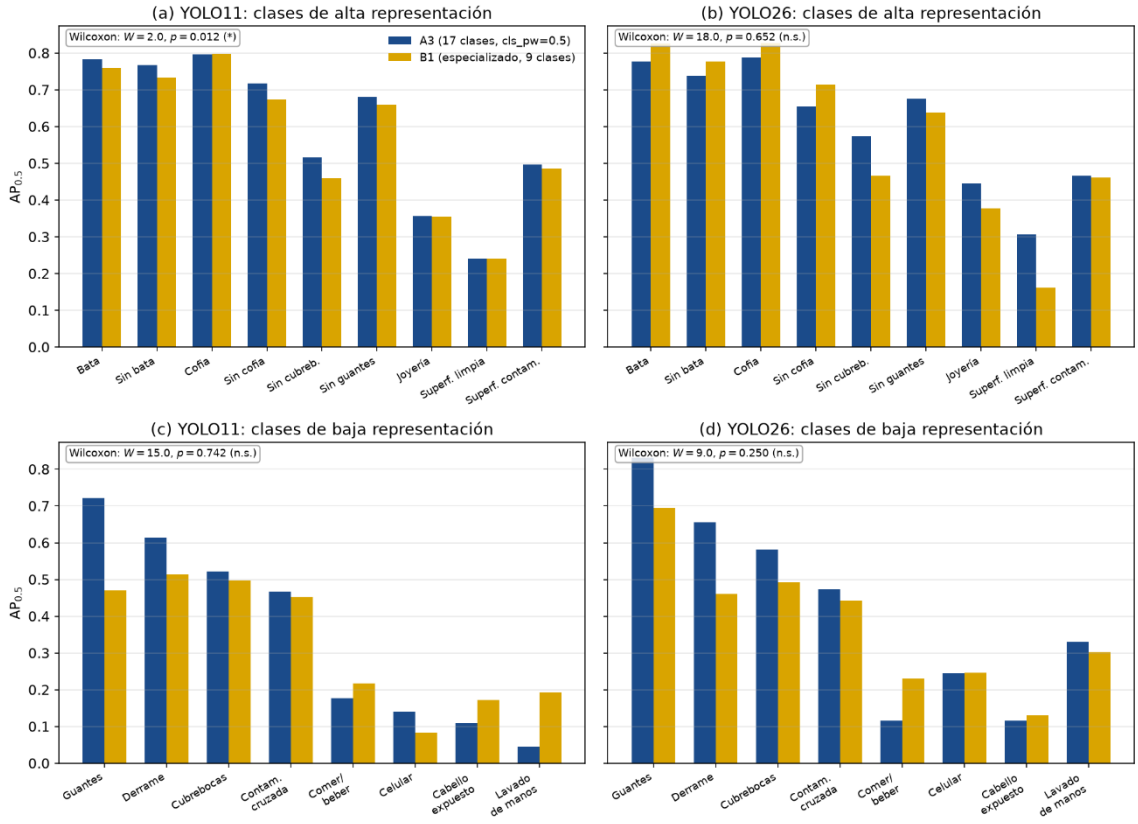


Figura 3. Comparación del AP50 por clase entre el sistema completo de 17 clases con ponderación por frecuencia inversa (A3, cls_pw=0.5, azul) y modelos especializados entrenados solo en un subconjunto de clases (B1/B2, dorado), para YOLO11 (a, c) y YOLO26 (b, d), ambos a tamaño 's'. Fila superior: 9 clases de alta representación (≥ mediana de instancias, Experimento B1). Fila inferior: 8 clases de baja representación (Experimento B2, la comparación que constituye el Experimento C). Los valores de Wilcoxon (prueba de rangos con signo, pareada por clase) se muestran en cada panel. Para las clases raras, ninguna arquitectura mostró una diferencia significativa entre especializar y ponderar (p=.742 y p=.250). Para las clases frecuentes, YOLO11 mostró una diferencia significativa a favor de mantener el sistema completo (p=.012), mientras que YOLO26 no (p=.652).

4.7 Evaluando el costo computacional y la latencia de inferencia

Se midió el tamaño, el costo computacional teórico (GFLOPs) y la latencia real de inferencia de las cinco arquitecturas bajo hardware idéntico:

Tabla 5. Comparación de precisión, tamaño y velocidad de inferencia por arquitectura

Arquitectura	mAP@0.5	Latencia	FPS	GFLOPs
YOLOv8	0.515	43.9 ms (la más rápida)	22.8	64.5
YOLOv10	0.455	49.7 ms	20.1	55.9

Arquitectura	mAP@0.5	Latencia	FPS	GFLOPs
YOLO26	0.511	88.4 ms	11.3	50.8
YOLO11	0.479	75.9 ms	13.2	48.6
YOLO12	0.500	106.2 ms (la más lenta)	9.4	48.5

YOLO12 y YOLO11 presentan un costo computacional teórico casi idéntico (48.5 frente a 48.6 GFLOPs), pero YOLO12 resultó 2.4 veces más lenta en inferencia real, lo que indica que el costo teórico no predice por sí solo la velocidad de ejecución. Esta diferencia es consistente con el tipo de operación empleada por cada arquitectura: los mecanismos de atención, utilizados de forma intensiva en YOLO12, permiten relacionar directamente información distante dentro de la imagen (Vaswani et al., 2017), lo que explica la ventaja de precisión de arquitecturas como YOLO26; sin embargo, estas operaciones son intrínsecamente menos eficientes en GPUs de consumo, que carecen de los aceleradores especializados para atención presentes en hardware de centro de datos (Dao et al., 2022).

Un segundo hallazgo, de naturaleza distinta, se observó durante el entrenamiento: en corridas aisladas de corta duración, YOLO12 entrenó a un ritmo normal (0.98 min/época). En una sesión extensa con las cinco arquitecturas entrenando en secuencia durante más de 40 horas, su configuración de mayor tamaño mostró un ritmo de ~19 min/época (un incremento de casi 20 veces), con la GPU al 100% de utilización y la memoria de video casi saturada. Se plantea como hipótesis, sin respaldo bibliográfico confirmado, que este comportamiento se debe a fragmentación de memoria acumulada tras un número elevado de entrenamientos consecutivos en el mismo proceso; esta hipótesis queda como pregunta abierta. Por esta razón YOLO12 se excluyó de la comparación final de A4.

4.8 Definiendo la arquitectura recomendada para despliegue

Para un sistema de inferencia local sobre cámara fija, donde la latencia es menos crítica que en un dispositivo embebido, YOLO26 ofrece el mejor equilibrio entre precisión y velocidad; YOLOv8 constituye la alternativa preferente si se prioriza la velocidad de inferencia ante un eventual escalamiento a múltiples cámaras.

5 Conclusiones

BPH-YOLO se sometió a un diseño experimental de siete bloques más un benchmarking de latencia, cubriendo cinco arquitecturas, tres estrategias de ponderación y dos configuraciones de especialización. La evidencia muestra que el desempeño del sistema depende de la interacción entre volumen de datos, desbalance de clases y arquitectura, y no de un solo factor: ni ponderar ni especializar mejoraron de forma significativa el desempeño sobre clases raras, y la comparación entre arquitecturas reveló tres perfiles funcionales distintos, sin una arquitectura dominante. YOLO11 obtuvo el mejor desempeño con dataset reducido; YOLO26 exige mayor volumen de datos y capacidad de modelo, pero supera a las demás una vez satisfechas esas condiciones; YOLOv8 ofrece el mejor rendimiento por unidad de tiempo de inferencia. Esta complementariedad de perfiles, más que una jerarquía fija entre arquitecturas, es uno de los hallazgos centrales del estudio, ya que la arquitectura óptima depende del estado de madurez del dataset disponible.

Por esto, YOLO11 se recomienda mientras el acervo de imágenes siga en expansión. Queda pendiente verificar si la ventaja de YOLO26 se sostiene o se incrementa al ampliar el dataset hacia cocinas industriales y laboratorios de docencia de alimentos, lo que definiría el punto de transición hacia esa arquitectura como estándar de producción. La ausencia de efecto de la ponderación y la especialización redirige la estrategia hacia la recolección dirigida de datos para clases raras, en lugar del ajuste de hiperparámetros. La degradación de YOLO12 bajo entrenamiento prolongado plantea, además, una pregunta metodológica relevante para cualquier comparación de arquitecturas bajo restricciones de cómputo.

La exclusión de YOLO12 de la comparación final responde a una restricción de tiempo y no a una decisión metodológica, por lo que no se descarta su viabilidad si se resuelve la inestabilidad observada en sesiones de entrenamiento prolongadas. Las pruebas estadísticas se calcularon sobre muestras reducidas (cinco arquitecturas, entre ocho y diecisiete clases), lo que limita su poder estadístico. El benchmarking de latencia, además, se realizó sobre un único tipo de GPU de consumo, de modo que los patrones de eficiencia observados, en particular el costo elevado de YOLO12, podrían no generalizar a hardware con aceleración de atención dedicada.

Como trabajo pendiente se plantean tres líneas: completar las combinaciones restantes de YOLO12 mediante procesos de entrenamiento aislados por arquitectura; ampliar el dataset hacia entornos reales de mayor escala y, con ello, reevaluar la conveniencia de migrar de YOLO11 a YOLO26 como arquitectura de producción; y avanzar hacia la Fase 2 del sistema de trackeo. Sobre este último punto, cabe destacar que la Fase 1, correspondiente al trackeo de personas mediante BoT-SORT con reidentificación, en la que cada incumplimiento detectado se asocia a una persona mediante Intersección sobre Unión, ya se encuentra en desarrollo y fue validada sobre 17,754 cuadros de video real de laboratorio, con reportes automáticos de inocuidad por persona. La Fase 2 consistirá en conectar ese mismo sistema a la cámara en vivo, quedando así el desarrollo un paso más cerca del despliegue completo.

Más allá de los resultados numéricos, este trabajo documenta y corrige tres errores metodológicos que, de no haberse detectado, habrían comprometido la validez de las métricas reportadas: fuga de datos por fotogramas correlacionados, paginación no determinista en el conteo de anotaciones, y selección de modelo por última época en lugar de por el punto de mejor ajuste real. Estos hallazgos evidencian que la validez de una comparación de arquitecturas depende tanto del diseño experimental como de la infraestructura que lo ejecuta.

El proyecto responde directamente a los objetivos planteados, entre ellos caracterizar el desbalance de clases, aislar el efecto del volumen de datos y la arquitectura, y validar un prototipo funcional de detección y trackeo. Asimismo, contribuye a tres Objetivos de Desarrollo Sostenible: el ODS 3 (salud y bienestar), al habilitar la detección temprana de incumplimientos de higiene que previenen enfermedades transmitidas por alimentos; el ODS 9 (industria, innovación e infraestructura), al ofrecer una solución de código abierto y bajo costo, replicable en instituciones educativas con recursos limitados; y el ODS 12 (producción y consumo responsables), al promover prácticas de manipulación de alimentos más seguras y verificables.

6 Anexos

El módulo de trackeo de personas (Fase 1 y 2, BoT-SORT con reidentificación por apariencia, descrito en las conclusiones) se encuentra implementado y disponible públicamente en un repositorio de GitHub bph-yolo-tracking. Este repositorio implementa la etapa de despliegue correspondiente al diseño experimental descrito en la Metodología (Bloques A1-C de este artículo) y sirve como base reproducible para que terceros puedan validar o replicar el protocolo de trackeo y asociación de incumplimientos aquí presentado. En congruencia con las salvaguardas de privacidad y manejo de datos descritas, no incluye el dataset anotado (imágenes de estudiantes), los modelos entrenados con datos privados, ni credenciales de acceso a los servicios utilizados. En caso de encontrar fallas de acceso o funcionamiento en el repositorio, se solicita contactar a la autora de correspondencia.

7 Bibliografía

AENOR. (s.f.). Norma ISO 22000 de seguridad alimentaria. https://www.aenor.com/certificacion/empresas/alimentacion/seguridad-alimentaria

Anthropic. (2026). Descripción general de Claude Code. Claude Code Docs. https://code.claude.com/docs/es/overview

DataCamp. (2026). YOLO object detection explained: A beginner's guide. https://www.datacamp.com/blog/yolo-object-detection-explained

Databricks. (s.f.). ¿Qué es una red neuronal? https://www.databricks.com/es/blog/what-is-neural-network

Diario Oficial de la Federación. (2010). NOM-251-SSA1-2009: Prácticas de higiene para el proceso de alimentos, bebidas o suplementos alimenticios. https://www.dof.gob.mx/normasOficiales/3980/salud/salud.htm

GoDaddy. (2024). Visual Studio Code: funcionalidades y beneficios para desarrolladores. https://www.godaddy.com/resources/latam/desarrollo/visual-studio-code-editor

Grahammiranda. (2026). Programación de vibe en 2026: guía completa, herramientas y modelos. https://tech.grahammiranda.com/es/vibe-coding-2026-guia-completa

IBM. (2025). ¿Qué son las redes neuronales convolucionales? https://www.ibm.com/mx-es/think/topics/convolutional-neural-networks

IBM. (2025). ¿Qué es el fine-tuning? https://www.ibm.com/es-es/think/topics/fine-tuning

IBM. (s.f.). ¿Qué es el vibe coding? https://www.ibm.com/es-es/think/topics/vibe-coding

International Organization for Standardization. (s.f.). ISO 22000:2018 Sistemas de gestión de la inocuidad de los alimentos. https://www.iso.org/es/contents/data/standard/06/54/65464.html

Javadex. (2026). Claude Code 2026: qué es, cómo funciona y para qué sirve. https://www.javadex.es/blog/claude-code-2026-que-es-como-funciona-anthropic

Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0) [Software]. Ultralytics. https://github.com/ultralytics/ultralytics

Jocher, G., & Qiu, J. (2024). Ultralytics YOLO11 (Version 11.0.0) [Software]. Ultralytics. https://github.com/ultralytics/ultralytics

KW Foundation. (s.f.). Redes neuronales convolucionales (CNN) vs. redes neuronales recurrentes (RNN). https://kwfoundation.org/blog/2021/07/13/redes-neuronales-convolucionales-cnn-vs-redes-neuronales-recurrentes-rnn/

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. arXiv. https://arxiv.org/abs/1405.0312

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. Biochemia Medica, 22(3), 276–282. https://doi.org/10.11613/bm.2012.031

NxCode. (2026). ¿Qué es el vibe coding? Guía completa para 2026. https://www.nxcode.io/es/resources/news/vibe-coding-guide-2026

OpenWebinars. (2024). Transfer learning: todo lo que necesitas saber. https://openwebinars.net/blog/transfer-learning/

Organización Mundial de la Salud. (2022). Inocuidad de los alimentos. https://www.who.int/es/news-room/fact-sheets/detail/food-safety

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788. https://doi.org/10.1109/CVPR.2016.91

Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of YOLO architectures in computer vision. Machine Learning and Knowledge Extraction, 5(4), 1680–1716. https://doi.org/10.3390/make5040083

Ultralytics. (s.f.). COCO detection dataset. Ultralytics Docs. https://docs.ultralytics.com/datasets/detect/coco

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). YOLOv10: Real-time end-to-end object detection. arXiv:2405.14458. https://doi.org/10.48550/arXiv.2405.14458

Willison, S. (2025, 19 de marzo). Not all AI-assisted programming is vibe coding (but vibe coding rocks). Simon Willison's Weblog. https://simonwillison.net/2025/Mar/19/vibe-coding/

Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. Advances in Neural Information Processing Systems, 35, 16344–16359.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association, 32(200), 675–701.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics Bulletin, 1(6), 80–83.