Comparativa del uso de redes neuronales convolucionales para reconocimiento artificial de sustancias en un laboratorio controlado.

A. A. Murillo Barrientos Instituto Politécnico Nacional, ESCOM. Unidad Profesional Adolfo López Mateos, Colonia Lindavista, Alcaldía Gustavo A. Madero, C.P. 07738, Ciudad de México E-mail: murilloalexis334@gmail.com, URL ORCID: https://orcid.org/0009-0001-3700-4561

A. Horta Hernández, Instituto Politécnico Nacional, ESCOM. Unidad Profesional Adolfo López Mateos, Colonia Lindavista, Alcaldía Gustavo A. Madero, C.P. 07738, Ciudad de México E-mail: andreahortahernandez@gmail.com URL ORCID: https://orcid.org/0009-0006-9795-1030

I. A. Alarcón Sánchez Instituto Politécnico Nacional, ESIME Zacatenco. Unidad Profesional Adolfo López Mateos, Colonia Lindavista, Alcaldía Gustavo A. Madero, C.P. 07738, Ciudad de México, E-mail: iaalarcons@gmail.com URL ORCID: https://orcid.org/ 0000-0003-0697-479X

Resumen *— En el campo de la visión por computadora, las redes neuronales YOLO (You Only Look Once), Mask R-CNN (Mask Region-Based Convolutional Neural Network) y EfficientDet son algunas de las arquitecturas más avanzadas para la detección de objetos. La elección entre YOLO, Mask R-CNN y EfficientDet depende de las necesidades específicas de la aplicación: YOLO es preferido para tareas en tiempo real, Mask R-CNN para detección y segmentación detallada, y EfficientDet para un equilibrio entre precisión y eficiencia computacional. En este trabajo de investigación se describen, de manera técnica y prácticas, las diferencias que existen entre dichas redes en un entorno especifico de uso, donde se demuestra que en rendimiento respecto a detección, YOLOv5 es mejor opción, pero sacrificando ratio detección mientras que Mask R-CNN tiene un mejor ratio de detección pero con rendimiento y tiempo de detección poco viables, para el caso de uso.*

Palabras Clave **– redes neuronales, YOLO, Mask R-CNN, EfficientDet, reconocimiento, entrenamiento**

Abstract -- In the field of computer vision, YOLO (You Only Look Once), Mask R-CNN (Mask Region-Based Convolutional Neural Network) and EfficientDet neural networks are some of the most advanced architectures for object detection. The choice between YOLO, Mask R-CNN and EfficientDet depends on the specific needs of the application: YOLO is preferred for real-time tasks, Mask R-CNN for detailed detection and segmentation, and EfficientDet for a balance between accuracy and computational efficiency. This research work describes, in a technical and practical way, the differences between these networks in a specific use case environment, where it is shown that in detection performance, YOLOv5 is a better choice, but sacrificing detection rate, while Mask R-CNN has a better detection rate but with unfeasible performance and detection time, for the use case.

Keywords **— Neural networks,YOLO, Mask R-CNN, EfficientDet, recognition, training.**

I. Introducción

Las redes neuronales convolucionales han demostrado ser herramientas poderosas en el campo de la visión por computadora, permitiendo el desarrollo de sistemas avanzados capaces de identificar y clasificar objetos con alta precisión. Sin embargo, cada arquitectura de red tiene sus propias características, ventajas y limitaciones que deben ser consideradas a la hora de implementarlas en aplicaciones prácticas.

Para los fines de este estudio, se han seleccionado y analizado tres de las arquitecturas más representativas y utilizadas en el ámbito del reconocimiento de imágenes: YOLOv5, Mask R-CNN y EfficientDet. Estas tres redes han sido entrenadas específicamente para reconocer ampolletas de laboratorios clínicos y clasificarlas según sus colores y tamaños. Esta tarea es de gran relevancia en entornos donde se manipulan sustancias clínicas o potencialmente peligrosas, ya que la correcta identificación y clasificación de estos elementos es crucial para garantizar la seguridad y la eficiencia en los procesos productivos y clínicos.

El funcionamiento adecuado de las redes neuronales empleadas en un sistema de este tipo es fundamental, ya que cualquier error en el reconocimiento puede representar riesgos significativos, tanto en términos de la seguridad del proceso productivo como para las personas involucradas. Por ello, es importante no solo comparar el rendimiento de estas redes en términos de precisión y velocidad, sino también considerar su aplicabilidad y robustez en un entorno clínico o industrial.

En este trabajo, se aborda la problemática de la búsqueda de una red neuronal que funcione con recursos computacionales limitados con un rendimiento lo más alto posible en sus pruebas de detección, por lo cual se describen en detalle las diferencias técnicas y prácticas de cada una de estas redes, incluyendo su capacidad de detección, tiempo de inferencia y facilidad de implementación. Además, se discuten las consideraciones específicas que deben tenerse en cuenta al elegir una arquitectura de red para aplicaciones sensibles, como la que se propone en este estudio.

Lo anterior responde a la problemática

II. Marco Teórico y Metodología

1. *Redes convolucionales*

Las redes neuronales de tipo convolucionales (CNN) son una de las diversas clasificaciones de redes neuronales que han demostrado en específico ser muy efectivas en áreas como el reconocimiento y la clasificación objetos en imágenes [1], dado a la capacidad de abstraer características que comparten objetos en distintas imágenes modelando una idea de qué es y qué debe cumplir algo, para clasificarse como un objeto.

Ahora bien, las CNNs trabajan en base como una red neuronal común, es decir, que el componente base de la red son, lo perceptrones, los cuales son la forma en como emulamos el comportamiento de una neurona humana por medio de matemáticas, en otras palabras, un perceptrón es una suma ponderada más un sesgo, y finalmente una función de activación (figura 1).

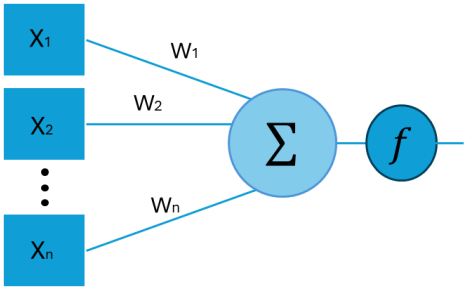

Figura 1. Modelo ilustrativo de un perceptrón.

Por otro lado, los perceptrones se juntan en capas, cada capa tiene determinada cantidad de perceptrones que comparten todos las mismas entradas y salidas [1].

Las capas se conectan entre sí, de tal manera que la entrada de una es la salida de otra, concretando modelos multicapa, en el caso de una CNN, la primera capa tiene la característica de tener conjuntos de neuronas que reciben como entrada una sección de la imagen a analizar que es leída como un tensor de grado 3, en otras palabras, se entiende a este tensor como tres matrices de dimensiones de ancho y largo iguales a las de la imagen, mientras que, el hecho que haya 3 matrices representan respectivamente la distribución RGB del pixel en dicha posición, cabe recalcar que en ocasiones la redes pueden ser hechas en base a tensores de grado 2 que significa que son imágenes en blanco y negro sin los canales de color rojo, verde y azul [2].

Luego que la imagen es seccionada en sectores, estos últimos son tratados por conjuntos de neuronas independientes a otros conjuntos, estas capas son las capas convolucionales que realizan, por medio de una matriz, que generalmente es de 3x3 denominada como kernel o filtro, un barrido por la sección donde por cada posición se realiza un producto escalar, obteniendo el Mapa de características [3], dado que el kernel debe de pasar por todos los pixeles que procesa la sección, se tiende a tener perdida de información en los bordes de las secciones, dado esto, es importante el agregar el padding, el cual es un margen de pixeles con valores que no afecten el cálculo del kernel, generalmente se usan valores como el cero[4]

Luego de una capa convolucional, una operación de activación ReLU(Rectified Linear Unit) que es una operación no lineal [2] se expresa como:

$$F(x) = \max(0, z) \tag{1}$$

La operación ReLU se encarga de incrementar la no linealidad de la CNN debido a que las imágenes que se obtiene del mundo real son no lineales y la CNN[2].

Además de las capas convolucionales, una CNN cuenta con capas intermedias denominadas polling, submuestreo o agrupación, dichas capas permiten reducir el costo computacional en memoria y procesamiento por medio de obtener los datos más significativo con un muestreo [3].

Las capas polling reducen las dimensiones de la sección que representan, por medio de seleccionar kernels o matrices de 2x2 y usar la función máxima(max) entre los 4 elementos que

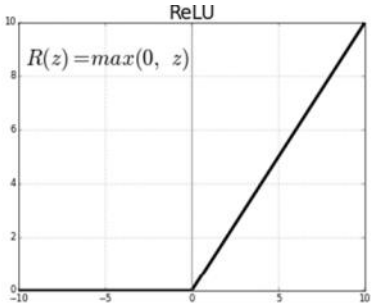

Figura 2. La opereción ReLU [1]

conforman el kernel, lo cual se traduce a un nuevo mapeado de los pixeles con dimensiones reducidas [5]

Cabe recalcar no solamente se utiliza la función máximo, sino que también se puede usar el promedio de los 4 pixeles del kernel.

Por último, luego de agregar múltiples capas convolucionales con función activación ReLU, y respectivas capas polling, se llega a as capas completamente conectadas, que serán las encargadas de hacer la función de clasificación [6], es decir, que se encarga de indicar los objetos que se detectan y con que tasa de porcentaje la red estima que es ese objeto.

La primera CNN fue la LeNet-5 la cual tenía 7 capas dividas en 3 convolucionales, 2 polling, y 2 completamente conectadas, con lo cual lograba detectar hasta 4 objetos [6].

2. *Ejemplos de redes convolucionales*

2.1. YOLO (You Only Look Once) version 7

La versión 7.0 de YOLO, destaca la introducción de nuevos modelos de segmentación que son los más rápidos y precisos disponibles en la actualidad, superando todos los benchmarks del estado del arte (SOTA). Estos modelos están diseñados para ser extremadamente fáciles de entrenar, validar y desplegar, similar a los modelos de detección de objetos existentes en YOLOv5.

Principales Actualizaciones:

- Modelos de Segmentación: Se han lanzado por primera vez modelos de segmentación YOLOv5-seg preentrenados en COCO, prometiendo mejoras continuas.

- Exportación a Paddle: Ahora es posible exportar cualquier modelo YOLOv5 (clasificación, segmentación o detección) al formato Paddle utilizando export.py.

- YOLOv5 AutoCache: Una nueva función que optimiza el uso de la memoria RAM durante el entrenamiento, reduciendo riesgos y acelerando el proceso de cacheo de datos.

- Integración con Comet: Esta herramienta gratuita permite guardar modelos YOLOv5, reanudar entrenamientos y visualizar interactivamente las predicciones.

Detalles Técnicos:

Los nuevos modelos de segmentación fueron entrenados en el conjunto de datos COCO durante 300 épocas con GPUs A100, y luego exportados a ONNX FP32 para pruebas en CPU y a TensorRT FP16 para pruebas en GPU. Todas las pruebas de velocidad se realizaron en Google Colab Pro, facilitando su reproducibilidad.

En total, esta versión incluye 280 pull requests de 41 colaboradores desde la última actualización en agosto de 2022. [7]

2.2. Mask-RCNN

Mask R-CNN, un marco conceptual simple, flexible y general para la segmentación de instancias de objetos en imágenes. Esta técnica extiende Faster R-CNN añadiendo una rama para predecir máscaras de objetos simultáneamente con la detección de objetos mediante cajas delimitadoras. Mask R-CNN se destaca por su simplicidad en el entrenamiento y su rápida ejecución (5 fps), añadiendo solo un pequeño sobrecoste a Faster R-CNN.

Mask R-CNN es versátil y fácil de generalizar a otras tareas, como la estimación de poses humanas, y ha logrado resultados superiores en tres desafíos del conjunto de datos COCO: segmentación de instancias, detección de objetos con cajas delimitadoras, y detección de puntos clave en personas. A pesar de ser un modelo relativamente simple, Mask R-CNN supera a todos los modelos anteriores en estas tareas, incluyendo a los ganadores del desafío COCO 2016.

El marco también introduce la capa RoIAlign, que mejora la precisión de las máscaras al evitar la cuantización espacial gruesa que utiliza RoIPool en Faster R-CNN. Este enfoque ha demostrado ser robusto, eficiente, y capaz de acelerar la investigación futura en el reconocimiento a nivel de instancia. El código de Mask R-CNN está disponible públicamente para la comunidad de investigación (figura 3). [8]

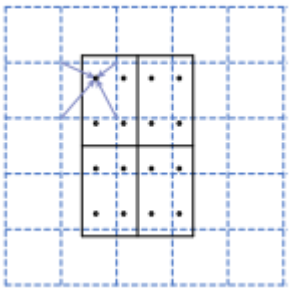

Figura 3 RolAlign [8]

2.3. EfficientDet

EfficientDet, una familia de detectores de objetos que se enfocan en optimizar la eficiencia del modelo, una necesidad creciente en la visión por computadora, especialmente para

aplicaciones en entornos con limitaciones de recursos como robótica y vehículos autónomos. A pesar de los avances recientes en la detección de objetos, los modelos de última generación han tendido a ser cada vez más grandes y costosos en términos de cálculo. Por ejemplo, detectores basados en NAS-FPN, como AmoebaNet, requieren una cantidad significativa de parámetros y operaciones de punto flotante (FLOPs), lo que limita su despliegue en aplicaciones del mundo real.

Contribuciones Clave:

1. BiFPN (Red Piramidal de Características Bidireccional Ponderada): EfficientDet introduce BiFPN, una red que facilita la fusión multiescala de características de manera rápida y eficiente. A diferencia de enfoques previos que simplemente suman las características de diferentes resoluciones, BiFPN introduce pesos ajustables que permiten que el modelo aprenda la importancia relativa de cada conjunto de características, aplicando repetidamente fusiones de arriba hacia abajo y de abajo hacia arriba. Esto resulta en una mejora significativa en la precisión sin sacrificar la eficiencia.

2. Escalado Compuesto: Otra innovación importante es el método de escalado compuesto, que permite escalar de manera uniforme la resolución, la profundidad y el ancho en todas las partes del modelo, incluyendo la red de predicción de caja/clase, la red de características, y el backbone. Este enfoque contrasta con trabajos anteriores que típicamente se enfocan en escalar solo el tamaño de la imagen o la complejidad del backbone, logrando un balance óptimo entre precisión y eficiencia.

3. Integración con EfficientNet: EfficientDet utiliza EfficientNet como backbone, conocido por su alta eficiencia en relación al tamaño del modelo y el número de operaciones de punto flotante. Al combinar EfficientNet con BiFPN y el escalado compuesto, EfficientDet logra un rendimiento superior en una amplia gama de restricciones de recursos.

EfficientDet ofrece mejoras sobresalientes en eficiencia y en comparación con detectores previos. Por ejemplo, el modelo EfficientDet-D7 alcanza un 55.1% de precisión promedio precisión (AP) en el conjunto de pruebas COCO test-dev, utilizando solo 77M de parámetros y 410B FLOPs, lo que es significativamente más eficiente que otros modelos líderes como YOLOv3, RetinaNet y detectores basados en NAS-FPN.

- Comparación de Rendimiento: Bajo las mismas restricciones de precisión, EfficientDet usa hasta 28 veces menos FLOPs que YOLOv3, 30 veces menos FLOPs que RetinaNet, y 19 veces menos FLOPs que NAS-FPN basado en ResNet. Esto se traduce en un modelo que es 4 a 11 veces más rápido en GPU/CPU que otros detectores anteriores.

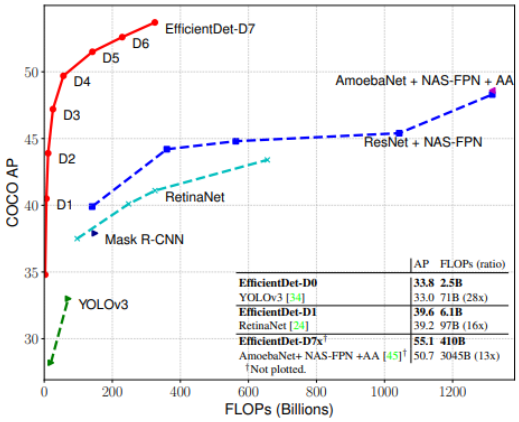

Figura 4 Model FLOPs vs. COCO accuracy [9]

- Rendimiento en Segmentación Semántica: EfficientDet también demuestra su capacidad en segmentación semántica, logrando un 81.74% de precisión mIOU en el conjunto de datos Pascal VOC 2012, superando a DeepLabV3+ en un 1.7% con 9.8 veces menos FLOPs (figura 4). [9]

3. *Redes convolucionales comparadas*

Para el caso de YOLO se utilizó la implementación preentrenada de YOLOv7m una versión de tamaño medio de la red, con EfficientNet, se utilizó la implementación preentrenada de EfficientDet-d3 hecha por Google® y finalmente para Mask-RCNN usamos igualmente una implementación con un modelo preentrenada.

El hecho que se haga el énfasis en que en cada caso se utilizó una implementación de la arquitectura preentrenada, es debido a las facilidades que ofrece, un modelo entrenado con un base de datos de miles o cientos de miles de imágenes permite tener un modelo fácil de adaptar al reconocimiento de ciertos objetos por medio de un nuevo conjunto de datos del orden de las centenas de imágenes, y usando menos recursos computacionales [10].

4. *Conjunto de imágenes y entrenamiento*

Se utilizo 350 imágenes de ampolletas de laboratorio con fondo variados, donde se procuró que todas las ampolletas, se tratan como una sola clase, aunque con variantes de colores, lo que lleva a que el modelo entrenado sea uniclase.

Las imágenes eran tratadas en un formato PNG para mantener la máxima cantidad de detalles, mientras que sus dimensiones son de 1600x900 pixeles.

Antes de ser cargas al modelo, se realizó un proceso de selección en cada imagen, donde se definía un área delimitada por un polígono que rodeaba a la ampolleta, lo cual genera un archivo ".txt" con las coordenadas de los vértices. Resaltando que el formato de coordenadas variaba dependiendo de la red.

Posteriormente se seleccionaron 75 imágenes del conjunto total, que servirían como validadoras, es decir, tiene el propósito qué en la etapa de entrenamiento, la red las utilice para verificar su fiabilidad de detectar el objeto

Para el entrenamiento de todas las redes, se utilizaron 80 épocas, mientras que los demás valores de entrenamiento se dejaron en valores por defecto.

Finalmente, para obtener los tiempos de procesamiento de por imagen, se realizó una media de cada imagen, este proceso se llevó a acabó por un script en Python que alimentaba al modelo constantemente con imágenes con el objeto a analizar, y guardabas las estadísticas que mandaba el modelo al procesar la imagen.

III. Resultados

Tabla 1. Resultados del entrenamiento

Redes Nauronales	Tiempo de procesamiento por imagen	(mAP) media promedio de precisión
EfficientDet-d0	31ms	0.832
YOLOv7m	20ms	0.885
Mask-RCNN	101ms	0.931

La tabla 1 indica de cada red neuronal; cuales fueron sus tiempos en milisegundos que tardo en procesar una imagen de entrada y determinar donde y con que ratio de detección creía que se encontraba la ampolleta.

Por otro lado, también se indica la precisión del modelo, esta estadística es proporcionada una vez se termina el entrenamiento de la red, he indica respecto a las imágenes de validación, cual fue el ratio de detección promedio, esta estadística se completo por medio del uso de imágenes extras usadas en el script de Python.

IV. Discusión

Aunque la arquitectura de la red Neuronal EffiencientNet en la literatura[9] mencione que es una excelente opción para sistemas con recursos limitados, tiene un gran problema al igual que Mask-RCNN, las implementaciones que existen en la actualidad de estas redes, tiene muy poco mantenimiento, lo que provoca que a la hora de trabajar con ellas en un entorno local, sea común el fallo con las dependencias y librerías que usan, los fallos van desde versiones deprecadas, hasta errores de compatibilidad, un problema que lleva a ser poco atractivas por su dificultad a la hora de trabajar con ellas, esto también lleva a que la red YOLO sea ampliamente usada, ya que se mantiene contantemente actualizada.

V. Conclusiones

Dado que el objetivo principal de esta investigación es analizar que red neuronal es mejor en términos de precisión de detección, coste computacional y rapidez de la detección.

Se puede concluir que Mask RCNN ha mostrado ser la mejor para este caso de uso, aunque es un modelo ciertamente antiguo al igual que EfficientDet, muestra un buen ratio de detección de objetos aunque acosta de un gran tiempo de procesamiento, que ampliamente rebasado por YOLO, aun así en términos de precisión gana Mask.RCNN, mientras que EfficientDet

et, tiene un buen desempeño en tiempo de detección, no justifica su bajo ratio de detección lo cual al final, en un laboratorio, es lo más importante.

Referencias

[1]. J. Sánchez-Alor Expósito, *Evaluación de algoritmos de detección de objetos basados en deep learning para detección de incidencias en carreteras* (Universidad de Valladolid, 2020).

[2]. J. Wu, *Introduction to Convolutional Neural Networks* (National Key Lab for Novel Software Technology, Nanjing University, China, 2017), p. 495.

[3]. S. Rozada Raneros, *Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning* (2021).

[4]. Z. Li, F. Liu, W. Yang, S. Peng y J. Zhou, 'A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects', *IEEE Transactions on Neural Networks and Learning Systems*,(2021),p 1–19. https://doi.org//// https://doi.org/10.1109/TNNLS.2021.3084827.

[5]. K. O'Shea y R. Nash, 'An Introduction to Convolutional Neural Networks', *arXiv preprint* (2015)

https://doi.org/10.48550/arXiv.1511.08458

[6]. Y. Pang, M. Sun, X. Jiang y X. Li, 'Convolution in Convolution for Network in Network', *IEEE Transactions on Neural Networks and Learning Systems*,(2017) https://doi.org/10.1109/TNNLS.2017.2676130

[7]. Zenodo, 'Zenodo', 22 noviembre 2022 https://zenodo.org/records/7347926 [último acceso: 18 agosto 2024].

[8]. **K. He, G. Gkioxari, P. Dollár y R. Girshick,** 'Mask R-CNN', *arXiv preprint*, (2018) https://doi.org/10.48550/arXiv.1703.06870.

[9]. M. Tan, R. Pang y Q. V. Le, 'EfficientDet: Scalable and Efficient Object Detection', CoRR, V. (2019) https://doi.org/10.48550/arXiv.1911.09070

[10]. R. Keshari, M. Vatsa, R. Singh y A. Noore, 'Learning Structure and Strength of CNN Filters for Small Sample Size Training', en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), p. 49–58. https://doi.org/10.48550/arXiv.1803.11405