Reportes de investigación

Entrenamiento de un modelo de inteligencia artificial para la detección de Vehículos Aéreos No Tripulados (VANTs)

Training an Artificial Intelligence Model for the Detection of Unmanned Aerial Vehicles

ADRIÁN STACUL, JOSÉ REYNOSO, SERGIO SALUZZI, MARTÍN MORALES y AGUSTÍN LACOMI
Laboratorio Técnicas Digitales, Departamento Electrónica Aplicada, Instituto de Investigaciones Científicas y Técnicas para la Defensa (CITEDEF), Argentina
astacul@citedef.gob.ar

Resumen

Este trabajo presenta el desarrollo y entrenamiento de un modelo de inteligencia artificial especializado en la detección de Vehículos Aéreos No Tripulados (UAV, por sus siglas en inglés) con la potencial finalidad de fortalecer capacidades de vigilancia y control de espacios aéreos. Para lograr dicho objetivo se implementó una técnica de *transfer learning*, utilizando la arquitectura YOLO (You Only Look Once), mediante un modelo preentrenado. Esta técnica involucra un proceso de *fine-tuning* con un *dataset* personalizado, el cual contiene más de 3.000 imágenes variadas de diferentes UAV. Este modelo fue entrenado con parámetros de 100 *epoch*s, un *batch size* de 8 y una resolución de 960 píxeles, y se logró una precisión mayor al 60% en la detección de aeronaves no tripuladas.

También se realizó una implementación en lenguaje Python que incluye un sistema de procesamiento en tiempo real, optimizado mediante arquitectura multihilo, que separa las tareas de captura de video con la inferencia del modelo. Con este método se lograron latencias inferiores a 20 milisegundos y velocidades de procesamiento superiores a 50 cuadros por segundo. El sistema se validó procesando videos de alta definición, demostrando su viabilidad para aplicaciones operacionales, donde la velocidad de respuesta es crítica para la identificación temprana de amenazas aéreas potenciales.

Los resultados obtenidos confirman la hipótesis inicial: el entrenamiento especializado de modelos de aprendizaje profundo mejora la capacidad de detección frente a modelos genéricos. Este trabajo presentado tiene el objetivo final de aplicar técnicas de inteligencia artificial a problemáticas específicas de la Defensa Nacional, con potenciales aplicaciones en sistemas de vigilancia activa y apoyo a la toma de decisiones operacionales.

Palabras clave: inteligencia artificial – Vehículos Aéreos No Tripulados – aprendizaje profundo – defensa nacional – vigilancia del espacio aéreo

Abstract

This paper presents the development and training of an artificial intelligence model specialized in the detection of Unmanned Aerial Vehicles (UAVs), with the aim of strengthening surveillance and airspace control capabilities. To achieve this objective, a transfer learning technique was implemented using the YOLO architecture, leveraging a pre-trained model. This technique involves a fine-tuning process with a custom *dataset* containing more than 3,000 varied images of different UAVs. The model was trained with parameters of 100 *epochs, a batch size* of 8, and a resolution of 960 pixels, achieving an accuracy greater than 60% in unmanned aircraft detection.

A Python-based implementation was also developed, including a real-time processing system optimized through a multithreaded architecture that separates video capture tasks from model inference. With this method, latencies below 20 milliseconds and processing speeds above 50 frames per second were achieved. The system was validated through the processing of high-definition videos, demonstrating its feasibility for operational applications where response speed is critical for the early identification of

potential aerial threats.

The results confirm the initial hypothesis: specialized training of deep *learning* models improves detection performance compared to generic models. The ultimate goal of this work is to apply artificial intelligence techniques to specific National Defense problems, with potential applications in active surveillance systems and operational decision-making support.

Keywords: Artificial Intelligence – Unmanned Aerial Vehicles – Deep *Learning* – National Defense – Airspace Surveillance

Introducción

El uso extensivo de Vehículos Aéreos No Tripulados en las últimas décadas ha transformado radicalmente el panorama operacional, tanto en el ámbito civil como en el contexto de la Defensa Nacional. Estos sistemas, caracterizados por su accesibilidad económica, facilidad operativa y amplia disponibilidad comercial, se utilizan actualmente en múltiples aplicaciones de defensa, como así también en sectores productivos y de servicios. Sin embargo, esta misma innovación tecnológica ha generado desafíos en materia de seguridad y defensa, cuando la operación de UAVs es de forma malintencionada o negligente.

Los sistemas tradicionales de vigilancia aérea mediante radares, utilizados para la detección de aeronaves tripuladas de mediano y gran porte, presentan limitaciones significativas cuando se intentan aplicar para la identificación de UAV tipo cuadricóptero, hexacópteros u otras plataformas aéreas de muy bajo porte. Estos vehículos no tripulados se caracterizan por su reducida sección transversal radar, velocidades de desplazamiento variables y capacidad operativa a bajas altitudes, resultando difíciles de detectar mediante sensores convencionales.

Las técnicas de aprendizaje profundo basadas en redes neuronales convolucionales han demostrado capacidades superiores para la detección y clasificación de objetos en condiciones variables. No obstante, los modelos de inteligencia artificial genéricos entrenados con conjuntos de datos públicos como COCO o ImageNet reconocen categorías de uso general (como pueden ser "personas", "perros", etc.) y no contemplan clases específicas como los UAV (Krizhevsky *et al.,* 2012; Lin *et al.,* 2014). Esta limitación fundamental motiva el desarrollo de modelos especializados mediante técnicas llamadas *transfer learning*, la cual permite adaptar arquitecturas neuronales previamente entrenadas a dominios específicos, sin requerir procesos de entrenamiento completos (Pan y Yang, 2010).

El presente trabajo aborda esta problemática mediante el desarrollo y entrenamiento de un modelo de inteligencia artificial especializado en la detección de UAVs, utilizando la arquitectura YOLO como base tecnológica (Redmon *et al.,* 2016). A diferencia de los modelos preentrenados, este desarrollo incorpora un proceso de fine-tuning sobre un *dataset* personalizado que contempla la variabilidad morfológica, condiciones de iluminación diversas y diferentes ángulos de observación característicos de los vehículos aéreos no tripulados. El objetivo es desarrollar un sistema capaz de identificar UAVs y validar el modelo mediante el procesamiento de video de alta resolución.

Marco teórico

1. *Transfer Learning y Fine-Tuning*

El *transfer learning* es una técnica de aprendizaje automático que permite aprovechar el conocimiento adquirido por un modelo neuronal durante su entrenamiento, en una tarea determinada para resolver una tarea diferente, pero relacionada. En el contexto de la visión por computadora, esta aproximación implica utilizar una red neuronal previamente entrenada sobre un *dataset* totalmente nuevo.

El proceso de *fine-tuning* consiste en ajustar los pesos o ponderaciones de un modelo preentrenado,

mediante un entrenamiento adicional sobre un *dataset* especializado. Durante este proceso, las capas iniciales de la red, que aprendieron a detectar características básicas como bordes, texturas y formas geométricas durante el entrenamiento previo, se mantienen relativamente estables, mientras que las capas superiores, responsables de características más específicas y abstractas, se adaptan progresivamente al nuevo dominio.

Este enfoque permite alcanzar resultados precisos con una menor cantidad de datos de entrenamiento y menor tiempo de procesamiento en comparación con un entrenamiento completo, reduciendo significativamente los recursos computacionales necesarios y acelerando los tiempos de desarrollo tecnológico.

2. Arquitectura YOLO para detección de objetos

YOLO (You Only Look Once) es una familia de algoritmos de detección de objetos que, a diferencia de métodos tradicionales que requieren múltiples pasadas sobre la imagen mediante ventanas deslizantes o regiones, esta arquitectura procesa la imagen completa en una única evaluación de la red neuronal, lo cual se traduce en velocidades de inferencia significativamente superiores.

Un *bounding box* es una anotación rectangular utilizada en visión artificial para indicar la ubicación espacial de un objeto dentro de una imagen o fotograma de video. El modelo YOLO divide la imagen de entrada en una cuadrícula, y cada celda de esta cuadrícula predice múltiples *bounding boxes*, junto con sus probabilidades de clase y niveles de confianza asociados. Cada *bounding box* se define mediante cinco parámetros fundamentales: las coordenadas del centro del cuadro, su ancho, su altura, y un *confidence score* que indica la probabilidad de que el cuadro efectivamente contenga un objeto de interés.

Esta arquitectura permite la detección simultánea de múltiples objetos en diferentes ubicaciones de la imagen, con un único paso de procesamiento.

3. *Datasets* y formato de anotación para YOLO

El entrenamiento de modelos de detección de objetos requiere *datasets* estructurados, donde cada imagen tiene asociado un archivo de anotaciones que especifican la ubicación y clase de los objetos presentes. Para la arquitectura YOLO, el formato estándar de anotaciones consiste en archivos de texto plano donde cada línea representa un objeto mediante cinco valores numéricos: el identificador de clase, seguido de las coordenadas normalizadas del centro del *bounding box* y sus dimensiones normalizadas.

La organización del *dataset* sigue una estructura jerárquica definida mediante un archivo de configuración en formato YAML, que especifica las rutas a las carpetas de entrenamiento y validación, junto con los nombres de las clases a detectar.

El rendimiento de un modelo de detección depende, en gran medida, de la calidad y heterogeneidad del *dataset* utilizado. La inclusión de imágenes capturadas con diversos ángulos de visión, condiciones de iluminación variadas y entornos geográficos diferenciados permite al modelo desarrollar capacidades de

generalización efectivas hacia situaciones no observadas durante el entrenamiento.

4. Métricas de evaluación en detección de objetos

La evaluación del desempeño de un modelo de detección de objetos se realiza mediante métricas estandarizadas que miden simultáneamente su capacidad para identificar correctamente los objetos y ubicarlos con precisión espacial en la imagen. La Intersección sobre Unión (IoU) cuantifica el grado de superposición entre el cuadro predicho por el modelo y el cuadro de verdad fundamental del objeto; valores elevados de IoU indican una localización espacial más precisa. La Precisión Media (AP) evalúa la relación entre detecciones correctas y detecciones totales realizadas por el modelo, considerando distintos niveles de confianza. La Precisión Media Promedio (mAP) corresponde al promedio de la AP calculada sobre todas las clases detectadas y constituye el principal indicador global de rendimiento del sistema.

Adicionalmente, se utilizan las métricas de precisión (proporción de detecciones correctas respecto del total de detecciones realizadas) y exhaustividad o *recall* (proporción de objetos reales correctamente detectados respecto del total de objetos presentes). La puntuación F1, definida como la media armónica entre precisión y *recall*, proporciona una medida del desempeño general del modelo.

En conjunto, estas métricas permiten determinar la eficacia del sistema tanto en la identificación como en la localización espacial precisa de los objetos dentro de la escena, ambos aspectos fundamentales para aplicaciones de vigilancia operacional, donde tanto las falsas alarmas como las detecciones omitidas tienen implicancias sobre la efectividad del sistema.

Metodología

1. Preparación del *dataset*

El conjunto final del *dataset* utilizado para el entrenamiento comprende 3.333 imágenes divididas en dos subconjuntos: 3.011 imágenes destinadas al entrenamiento y 322 imágenes reservadas para validación, manteniendo una proporción aproximada de 90/10 considerada adecuada para evitar un sobreajuste y, a la vez, maximizar la capacidad de aprendizaje del modelo.

Cada imagen del *dataset* fue anotada manualmente utilizando herramientas especializadas que permiten delimitar los UAVs presentes mediante *bounding box*es rectangulares. Estas anotaciones se almacenaron en formato texto plano siguiendo la especificación estándar de YOLO. El *dataset* incluye una única clase denominada "UAV" que engloba diferentes morfologías de aeronaves no tripuladas, desde cuadricópteros hasta plataformas de ala fija de pequeño porte.

La estructura del *dataset* fue definida mediante un archivo de configuración YAML que especifica las rutas absolutas a las carpetas de entrenamiento y validación. Este archivo actúa como punto de entrada único para el proceso de entrenamiento y centraliza la información sobre la organización de los datos y los

parámetros de configuración del modelo.

2. Configuración del entrenamiento

El proceso de entrenamiento se realizó utilizando el *framework Ultralytics* YOLO versión 8, específicamente la variante YOLOv8m, que representa una arquitectura de capacidad intermedia dentro de la familia YOLOv8, que logra un equilibrio entre rendimiento y eficiencia computacional (Terven y Córdova-Esparza, 2023; Ultralytics, 2025). El modelo base cuenta con 25.856.899 parámetros distribuidos en 169 capas que incluyen capas convolucionales, bloques C2f, módulos SPPF y una cabeza de detección multi-escala. El Cuadro 1 presenta los parámetros principales de configuración del entrenamiento implementado.

Cuadro 1: configuración de parámetros del entrenamiento

Parámetro	Valor
MODEL_PATH	yolov8m.pt
DATASET_YAML	uav_dataset.yaml
*EPOCH*S	100
BATCH_SIZE	8
IMG_SIZE	960
PATIENCE	150
OPTIMIZER	AdamW
*LEARNING*_RATE	0.001
LRF	0.01
MOMENTUM	0.937
WEIGHT_DECAY	0.0005
COS_LR	True

El entrenamiento se configuró con un *batch size* de 8 imágenes procesadas simultáneamente, por limitaciones de memoria disponible en la GPU. La resolución de entrada se estableció en 960x960 píxeles, tamaño optimizado para modelos YOLO que permite detectar objetos de diversos tamaños manteniendo mayor nivel de detalle para objetos pequeños. El número total de *epochs* completados sobre el *dataset* se configuró en 100 iteraciones.

El optimizador seleccionado fue AdamW (*Adaptive Moment Estimation with Weight Decay*) con tasa de aprendizaje inicial de 0.001, factor de reducción final de 0.01 y *momentum* de 0.937, parámetros optimizados para el entrenamiento de redes de detección de objetos (Kingma y Ba, 2015; Loshchilov y Hutter, 2019). El entrenamiento se ejecutó sobre infraestructura equipada con GPU NVIDIA GeForce RTX 3060 de 12GB.

3. Sistema de detección en tiempo real

Una vez completado el entrenamiento, el modelo resultante fue integrado en un sistema de procesamiento

de video en tiempo real, diseñado para maximizar el rendimiento mediante una arquitectura multihilo. Este sistema implementa el patrón productor-consumidor donde un hilo dedicado se encarga exclusivamente de la captura de cuadros desde la fuente de video, mientras que un segundo hilo ejecuta la inferencia del modelo de detección sobre los cuadros capturados. El Cuadro 2 presenta la arquitectura simplificada del sistema de detección implementado.

Cuadro 2: arquitectura del sistema de detección en tiempo real

1	import cv2 import threading from queue import Queue from ultralytics import YOLO

2	# Cargar modelo entrenado
	model = YOLO("best.pt")

3	# Colas para comunicación entre hilos
	frame_queue = Queue(maxsize=5)
	result_queue = Queue(maxsize=5)

4	def video_reader(cap, queue):
	"""Hilo productor: captura cuadros del video"""
	while True:
	ok, frame = cap.read()
	if not ok:
	queue.put(None)
	break
	queue.put(frame)

5	def detector_worker(model, in_queue, out_queue):
	"""Hilo consumidor: ejecuta inferencia del modelo"""
	while True:
	frame = in_queue.get()
	if frame is None:
	out_queue.put(None)
	break

6	# Ejecutar detección
	results = model.predict(
	source=frame,
	imgsz=640,
	conf=0.15,
	verbose=False)
	out_queue.put((frame, results))

7	# Iniciar sistema
	cap = cv2.VideoCapture("video.mp4")
	threading.Thread(target=video_reader, args=(cap, frame_queue)).start()
	threading.Thread(target=detector_worker, args=(model, frame_queue,
	result_queue)).start()

8	# Procesar resultados
	while True:
	result = result_queue.get()
	if result is None:
	break
	frame, detections = result
	# Visualizar detecciones
	cv2.imshow("Detección UAV", frame)

Esta arquitectura multihilo permite que los procesos operen de manera concurrente, evitando que la latencia de la inferencia del modelo afecte la captura continua del *stream* de video. Los cuadros capturados se almacenan en una cola con mecanismo de sincronización *thread-safe* y capacidad máxima de 5 elementos. El hilo detector extrae cuadros de esta cola y ejecuta el método de predicción del modelo YOLO, configurado con un umbral de confianza de 0.15 y tamaño de procesamiento de 640 píxeles. Los resultados de cada detección, incluyendo coordenadas de *bounding box*es, niveles de confianza y clases detectadas, se visualizan en tiempo real mediante superposición sobre los cuadros originales, proporcionando una retroalimentación visual inmediata al operador del sistema.

Resultados

4. Métricas de desempeño

El modelo fue entrenado durante 100 *epoch*s completos, proceso que requirió 7,374 horas (aproximadamente 265 segundos por *epoch*). El *framework* automáticamente identifica y preserva el modelo con mejor desempeño en validación. Las métricas del mejor modelo obtenido son: precisión de 0.624, *recall* de 0.397, mAP50 de 0.413 y mAP50-95 de 0.183. Estos resultados indican que el modelo logra identificar correctamente un 62.4% de las detecciones realizadas, mientras que detecta aproximadamente el 39.7% de todos los UAVs presentes en las imágenes de validación. El mAP50 de 0.413 indica que, considerando un umbral de IoU de 0.5, el modelo alcanza una precisión promedio superior al 41% en la tarea de detección. Estos valores son útiles en la detección de objetos pequeños y morfológicamente diversos operando en entornos visuales complejos.

5. Validación operacional y rendimiento del sistema

El modelo entrenado fue integrado en el sistema de detección en tiempo real y validado procesando un video de alta definición de 15.382 cuadros, capturado mediante plataforma UAV. El sistema demostró capacidad para mantener operación estable con rendimiento de 52,24 cuadros por segundo y latencias de inferencia promedio de 19,05 milisegundos cuando se ejecuta sobre GPU dedicada. Esta velocidad de procesamiento resulta ampliamente compatible con aplicaciones de tiempo real donde se requiere respuesta inmediata ante la detección de amenazas aéreas potenciales.

Durante la validación del video completo, que requirió 294,47 segundos de procesamiento, se registraron 3.985 detecciones distribuidas en 3.814 cuadros, representando aproximadamente el 24,8% del total de cuadros procesados. El *confidence score* promedio de las detecciones fue de 0,321 (con umbral configurado en 0.15), con rango desde 0,151 hasta 0,901.

La velocidad de inferencia lograda (19,05ms promedio por imagen) permite el procesamiento en tiempo real, requisito fundamental para su viabilidad operacional en sistemas de vigilancia activa.

6. Conclusiones

El presente trabajo ha demostrado la viabilidad técnica de desarrollar sistemas de detección automática de UAVs especializados mediante la aplicación de técnicas de *transfer learning* a arquitecturas neuronales de detección de objetos. El modelo desarrollado alcanzó métricas de mAP50 de 41,3% y precisión de 62,4%, valores que resultan operacionalmente útiles considerando las características particularmente desafiantes inherentes a la detección de objetos pequeños y morfológicamente diversos operando en entornos visuales complejos. La arquitectura YOLOv8m seleccionada demostró ser un balance adecuado entre capacidad de detección y eficiencia computacional, con 25,8 millones de parámetros que permiten su despliegue en infraestructura accesible.

El alcance del mejor desempeño en el *epoch* 26 del entrenamiento valida la efectividad de la estrategia

de *transfer learning*, combinada con el optimizador AdamW, demostrando que los pesos preentrenados en tareas de visión general proporcionan una base de conocimiento que puede adaptarse eficientemente a dominios específicos de aplicación en defensa, incluso cuando se dispone de *datasets* de tamaño limitado. La velocidad de procesamiento lograda, con latencias de 19,05 milisegundos y capacidad superior a 52 cuadros por segundo en GPU, confirma la viabilidad técnica del sistema para aplicaciones de tiempo real donde la detección temprana constituye un factor crítico para la efectividad operacional.

El desarrollo tecnológico presentado contribuye al fortalecimiento de las capacidades nacionales en el área de vigilancia y control de espacios aéreos estratégicos, ámbito de creciente relevancia para la defensa nacional. El sistema implementado demuestra que es posible desarrollar capacidades autóctonas de detección automática de UAVs con rendimiento operacional, utilizando infraestructura computacional accesible.

Trabajo futuro

Las principales líneas de investigación y desarrollo futuro identificadas se orientan hacia el fortalecimiento de las capacidades del sistema mediante dos ejes complementarios: la mejora sustancial de la calidad del *dataset* de entrenamiento y la exploración de su implementación en plataformas de *hardware* embebido para aplicaciones de campo.

En relación a la construcción de un *dataset* especializado de mayor calidad, resulta fundamental enfocarse no en la mera expansión cuantitativa del número de imágenes, sino en la mejora cualitativa de las condiciones de captura y la diversidad de escenarios representados.

La diversidad morfológica de los UAVs representados debe incluir diferentes configuraciones de multirrotores (cuadricópteros, hexacópteros, octocópteros), sistemas de ala fija de diversos tamaños, y plataformas híbridas, capturados desde múltiples ángulos de observación y distancias variables.

La segunda línea de investigación futura consiste en el estudio de la implementación del sistema de detección en plataformas de *hardware* embebido. Los resultados obtenidos demuestran viabilidad operacional en infraestructura con GPU de escritorio, pero para aplicaciones de vigilancia distribuida es necesario explorar la portabilidad del sistema a plataformas integradas.

La implementación en *hardware* embebido es de suma utilidad para plataformas móviles terrestres, marítimas o aéreas, permitiendo capacidades de vigilancia distribuida que no dependan de infraestructura de procesamiento centralizada. Esta línea de desarrollo contribuiría a la optimización de modelos de inteligencia artificial en plataformas de recursos limitados aplicadas a la vigilancia y control de espacios aéreos nacionales.

Agradecimientos

Los autores expresan su reconocimiento a las autoridades y dirección del Instituto de Investigaciones Científicas y Técnicas para la Defensa (CITEDEF), que contribuyen al fortalecimiento de las capacidades nacionales en tecnologías aplicadas a la Defensa Nacional.

También un especial agradecimiento al Departamento de Mecánica Aplicada y al Departamento de Electrónica Aplicada del CITEDEF, que proporcionaron el apoyo necesario para el desarrollo de esta investigación.

Bibliografía

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934.

Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580-587) (CVPR).

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778) (CVPR).

Kingma, D. P., Ba, J. (2015). Adam: A Method for Stochastic Optimization. En *International Conference on Learning Representations (ICLR*).

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. En *Advances in Neural Information Processing Systems (NIPS)* (pp. 1097-1105), 25.

Lin, T.-Y., Maire, M., Belongie, S. *et al.* (2014). Microsoft COCO: Common Objects in Context. En *European Conference on Computer Vision (ECCV)* (pp. 740-755).

Loshchilov, I., Hutter, F. (2019). Decoupled Weight Decay Regularization. En *International Conference on Learning Representations (ICLR)*.

Pan, S. J., Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345-1359.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779-788).

Ren, S., He, K., Girshick, R., Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(6), 1137-1149.

Terven, J., Córdova-Esparza, D.-M. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction, 5*(4), 1680-1716.

Ultralytics (2025). YOLOv8 Documentation. Disponible en https://docs.ultralytics.com/. Consulta: 15 de octubre de 2025.