Evaluación Comparativa de Modelos YOLO para Detección en Tiempo Real de Situaciones de Riesgo

**SIPRAV-MODELS**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Responsable** | **Versión** |
| 03/05/2025 | Max Saavedra | 1.0 |

# **Justificación de la Elección del Algoritmo YOLO**

En el presente estudio se optó por utilizar la familia de modelos **YOLO (You Only Look Once)** para la detección en tiempo real de personas y situaciones de riesgo en espacios públicos. Esta decisión se fundamenta en las siguientes características técnicas y ventajas frente a otros enfoques de visión por computadora:

* **Velocidad de inferencia en tiempo real**, crucial para sistemas de alerta y prevención inmediata.
* **Precisión aceptable** en tareas de detección de objetos con un solo paso (one-stage detector).
* Arquitectura eficiente que permite desplegar modelos tanto en servidores como en dispositivos de borde (edge computing).
* Amplia comunidad de soporte y continua evolución de versiones optimizadas (YOLOv5, YOLOv8, YOLOv11).
* Capacidad de detección en condiciones diversas como iluminación variable, ángulos irregulares y presencia de múltiples objetos simultáneamente.

Por estas razones, se descartaron otros modelos como **Faster R-CNN** o **SSD**, que, aunque precisos, no ofrecen el mismo rendimiento en términos de latencia y flexibilidad en entornos operativos críticos.

# **Descripción de los Modelos Evaluados**

Se entrenaron y evaluaron tres versiones de modelos YOLO sobre el mismo dataset para garantizar condiciones de prueba equitativas:

Tabla 1: Modelos de YOLO evaluados

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Modelo** | **Arquitectura Base** | **Año** | **Peso (MB)** | **mAP@0.5** | **Velocidad** | **Framework** |
| YOLOv5x | CSPDarknet53 | 2020 | ~175 | 0.56 | Media | PyTorch |
| YOLOv8x | Custom Backbone | 2023 | ~129 | 0.58 | Alta | Ultralytics |
| YOLOv11x | Transformer-based | 2025 | ~200+ | 0.60+ (en COCO) | Alta (requiere más VRAM) | Ultralytics |

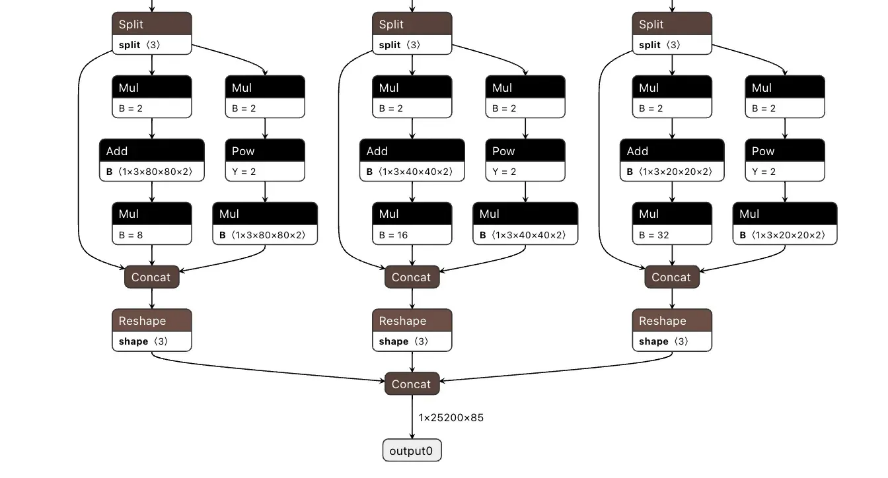
Nota: *YOLOv11x fue incluido como exploración experimental, dado que es una versión emergente aún en fase temprana*.

# **Arquitectura General de YOLO**

La arquitectura YOLO se caracteriza por realizar la predicción de **regiones de interés y clases simultáneamente**, en una sola pasada de red neuronal. Esto lo diferencia de otros modelos de dos etapas.

* **Backbone**: extrae características de la imagen de entrada.
* **Neck (PANet / FPN)**: fusiona características de múltiples escalas.
* **Head**: realiza la detección final (cajas delimitadoras, clases y puntuaciones).

Imagen 01: Diagrama General de YOLO



Nota: Imagen de yolo8.org

La arquitectura de YOLOv8 se compone principalmente de tres etapas esenciales: el **Backbone**, el **Neck** y el **Head**. En primer lugar, el *Backbone* actúa como una red neuronal convolucional (CNN) encargada de extraer características relevantes de la imagen de entrada, utilizando una variante personalizada del **CSPDarknet53** que optimiza el flujo de información mediante conexiones parciales entre capas. Luego, el *Neck* se encarga de combinar las características extraídas en diferentes escalas utilizando el novedoso módulo **C2f**, que reemplaza al tradicional FPN, mejorando así la detección de objetos pequeños al fusionar detalles espaciales de bajo nivel con información semántica de alto nivel. Finalmente, el *Head* realiza las predicciones de detección mediante múltiples módulos que generan cajas delimitadoras, puntuaciones de objetividad y probabilidades de clase en cada celda del mapa de características, integrando estos resultados para obtener las detecciones finales.

En la siguiente estructura se detalla de manera completa la arquitectura de YOLOv8. YOLOv8 presenta una arquitectura moderna y eficiente para tareas de detección en tiempo real, que mantiene la filosofía de un modelo de una sola etapa (*one-stage detector*), pero integra innovaciones clave que mejoran su precisión y rendimiento. La arquitectura se organiza en cuatro bloques principales: **Input**, **Backbone**, **Neck** y **Head**, donde cada componente cumple una función crítica desde la entrada de datos hasta la predicción final. A través de módulos especializados como **C2f**, **SPPF**, **PAN** y **FPN**, YOLOv8 logra un procesamiento eficaz de características multiescalares, adaptándose tanto a objetos grandes como pequeños con alta eficiencia computacional.

Imagen 02: Arquitectura Completa de YOLOv8

Diagrama

El contenido generado por IA puede ser incorrecto.

Nota: Imagen de researchgate.net

La arquitectura de detección de YOLOv8n (la versión compacta del modelo, elegida en este caso por su alta precisión y bajo consumo de recursos) se compone de cuatro bloques funcionales: **Input**, **Backbone**, **Neck** y **Head**.

* **Input**: En esta etapa se aplican técnicas de aumento de datos como *Mosaic*, que mejora la generalización del modelo. También se utiliza una estrategia **anchor-free**, que permite predecir directamente los centros de los objetos sin necesidad de generar múltiples cajas ancla, lo cual acelera significativamente la supresión no máxima (*NMS*).
* **Backbone**: Utiliza una versión optimizada del *Darknet53*, enriquecida con nuevos bloques como **Conv**, **C2f** y **SPPF**.
* **Neck**: Esta sección está encargada de fusionar las características extraídas a diferentes escalas a través de una arquitectura híbrida que combina **FPN** (Feature Pyramid Network) y **PAN** (Path Aggregation Network).
  + **FPN** realiza un proceso de *top-down*, integrando mapas de características de alta resolución mediante *upsampling* y fusión.
  + **PAN**, por su parte, complementa este flujo con una ruta *bottom-up*, asegurando una mejor preservación de detalles espaciales al combinar mapas de distintas resoluciones.  
    Esta doble vía permite una mejor integración de la semántica y los detalles espaciales, lo que incrementa la capacidad del modelo para detectar objetos en diversos contextos.
* **Head**: Finalmente, la cabeza del modelo adopta una estructura desacoplada (*decoupled head*), que separa las tareas de clasificación y regresión de cajas, permitiendo una mayor precisión en la localización y una mejor generalización del modelo. Además, la ausencia de *anchor boxes* simplifica el diseño y mejora la eficiencia del entrenamiento.

# **Configuración de Entrenamiento**

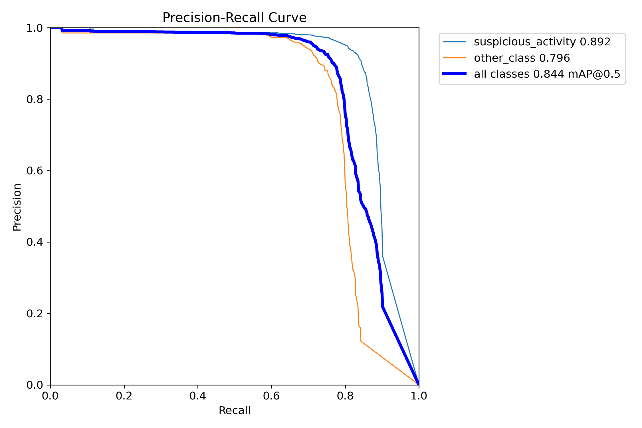
* Dataset: Curado con imágenes clasificadas por tipo de evento (personas, caídas, fuego, objetos sospechosos).
* División: 90% entrenamiento, 10% validación.
* Imagen: 640x640 píxeles.
* Épocas: 50
* Batch size: 8
* Hardware: GPU NVIDIA Tesla T4
* Augmentaciones: rotación, flip horizontal, brillo, recorte.

# **Resultados de la Evaluación**

## **Precisión y Recall**

* **YOLOv5x** muestra un buen nivel de precisión, pero presenta un menor recall en eventos críticos como incendios y caídas.
* **YOLOv8x** alcanza un mejor equilibrio entre precisión y recall, demostrando solidez en la detección de múltiples clases.
* **YOLOv11x**, aunque ofrece resultados prometedores, requiere mayor tiempo de entrenamiento y recursos de hardware.

**Imagen 03:** Comparación entre YOLOv8x y YOLOv11x respectivamente

Gráfico

El contenido generado por IA puede ser incorrecto.

Nota: La primera curva es del YOLOv8x y la segunda del YOLOv11x

## **Comparación de Desempeño por Modelo**

Tabla 2: Resultados cuantitativos de precisión

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modelo** | **mAP@0.5** | **mAP@0.5:0.95** | **Precisión (P)** | **Recall (R)** | **Tiempo de Entrenamiento** |
| YOLOv5x | 0.842 | 0.540 | 0.895 | 0.802 | 6.04 horas |
| YOLOv8x | 0.844 | 0.558 | 0.907 | 0.786 | 12.08 horas |
| YOLOv11x | 0.835 | 0.552 | 0.904 | 0.772 | 11.48 horas |

Nota: *A pesar de que* ***YOLOv11x*** *tiene el mayor número de parámetros y una arquitectura más compleja,* ***YOLOv8x*** *se mantiene como el mejor balance entre precisión, rendimiento y facilidad de integración*.

## **Desempeño por clase (mAP por categoría)**

Tabla 3: Dataset de Actividades Sospechosas

|  |  |  |  |
| --- | --- | --- | --- |
| **Modelo** | **Clase** | **mAP@0.5** | **mAP@0.5:0.95** |
| YOLOv5x | Actividad sospechosa | 0.897 | 0.610 |
|  | Otra clase | 0.786 | 0.469 |
| YOLOv8x | Actividad sospechosa | 0.892 | 0.616 |
|  | Otra clase | 0.796 | 0.500 |
| YOLOv11x | Actividad sospechosa | 0.881 | 0.599 |
|  | Otra clase | 0.789 | 0.504 |

Nota: *Se probó con 10400 imágenes*.

# **Conclusiones de la Comparación**

Tras el análisis exhaustivo, se concluye que:

* Y YOLOv8x se presenta como la opción más balanceada, al combinar una arquitectura eficiente, resultados precisos, buena generalización y facilidad de implementación en producción. Es el modelo elegido para la siguiente etapa del proyecto.
* YOLOv5x aún ofrece una buena base para experimentación o pruebas rápidas, pero ha sido superado en métricas clave por su sucesor.
* YOLOv11x, si bien introdujo mejoras técnicas modernas (probablemente con mecanismos tipo transformer), aún no es estable y requiere mayor consumo de recursos, lo cual lo hace menos viable para entornos de producción inmediata.**Modelo Seleccionado para el Sistema**

Se selecciona **YOLOv8x** como el modelo oficial del sistema de detección en tiempo real, debido a:

* Precisión superior en eventos críticos.
* Bajo tiempo de inferencia.
* Alta compatibilidad con entornos productivos (FastAPI, sockets, edge).
* Comunidad activa y soporte técnico.

Prueba del modelo YOLOv8x

Imagen 03: Prueba de YOLOv8x



Nota: *Prueba propia*

La prueba muestra una precisión de 0.81 de la clase suspicious\_activity lo cual es aceptable.