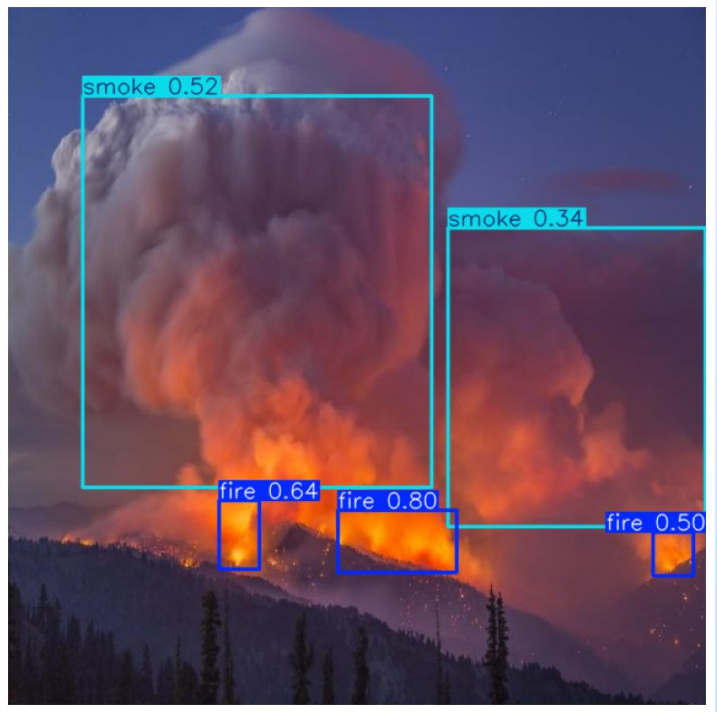
Visión por computadora II

Alumnos:

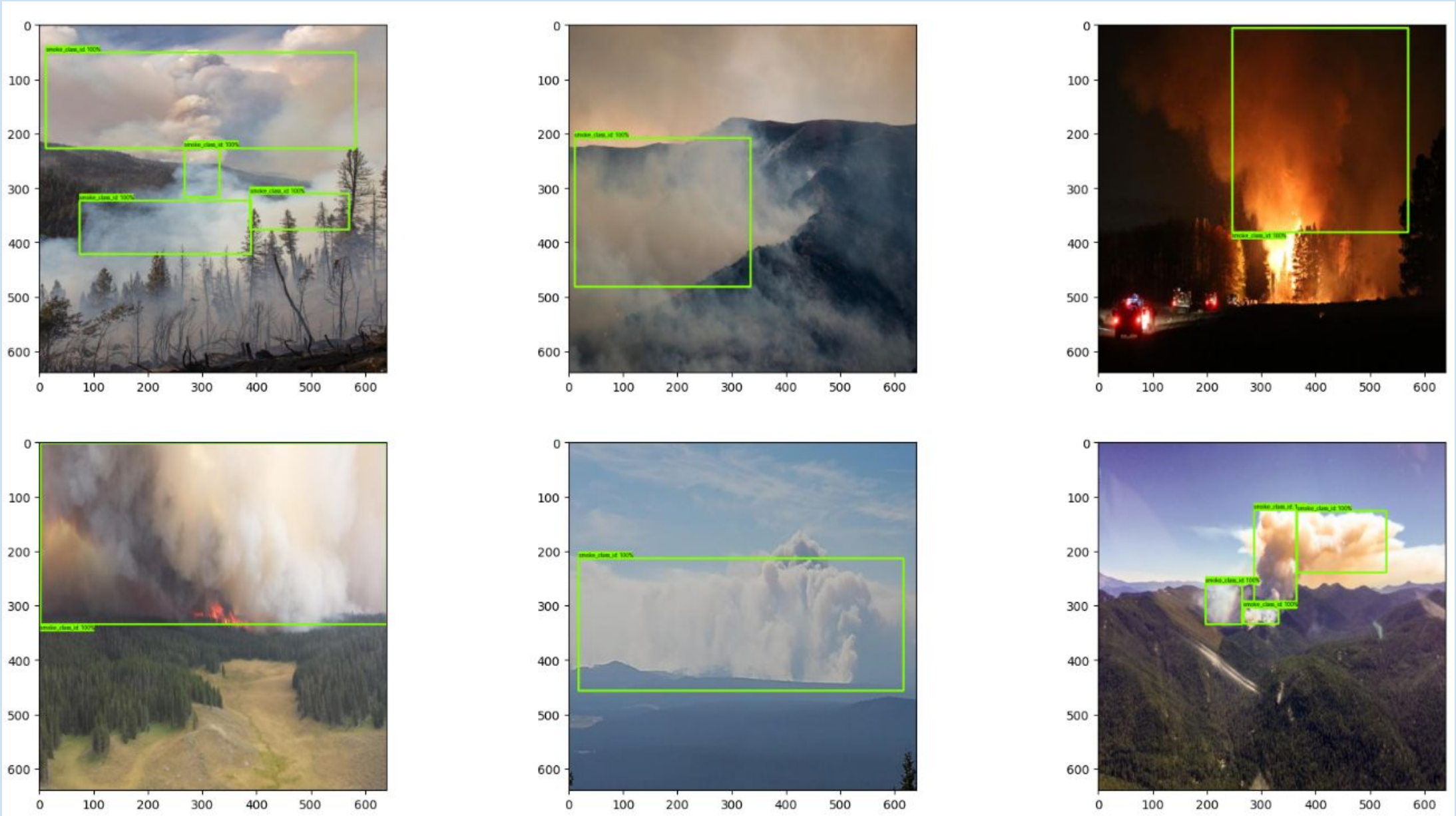
- Federico Arias Suárez
- Myrna Lorena Degano
- Gustavo Julián Rivas

Objetivo: Detectar focos de incendio.



Motivación: Los incendios en la Amazonía están incrementándose, afectando el clima, la biodiversidad y agravando el calentamiento global. La detección temprana de estos incendios es vital para prevenir daños mayores.

Dataset

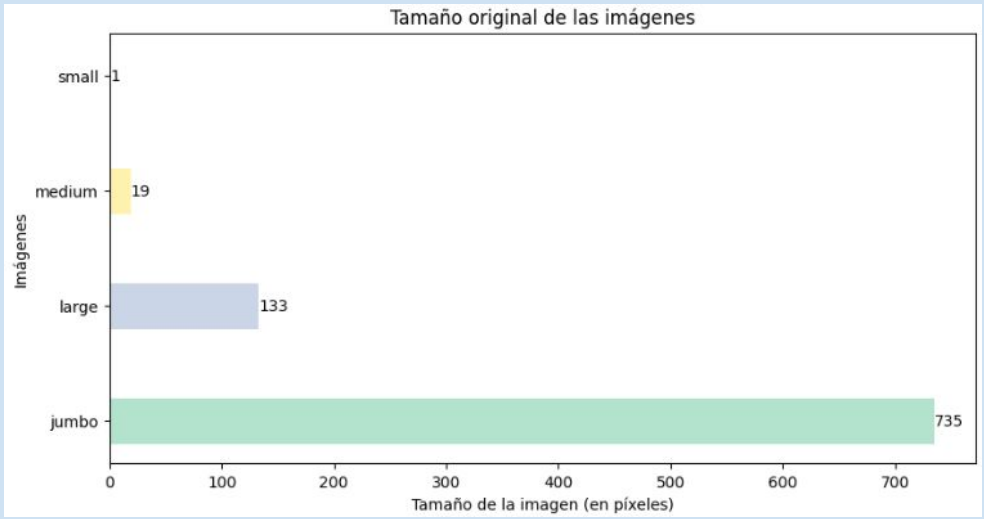


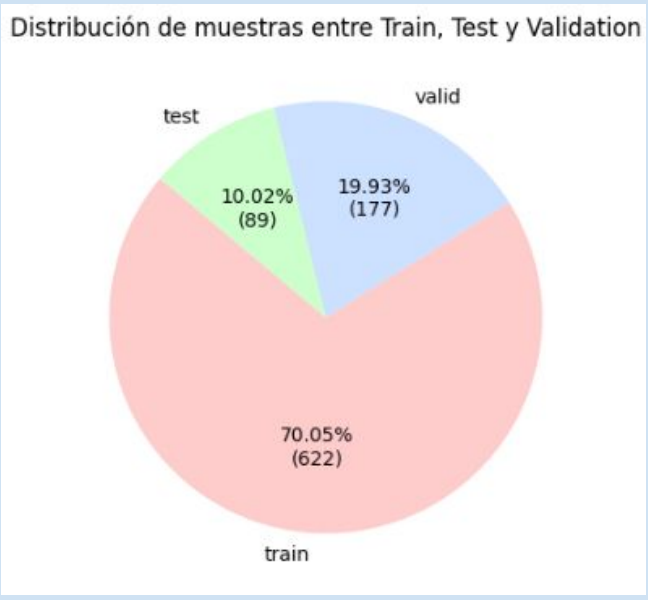
Análisis exploratorio

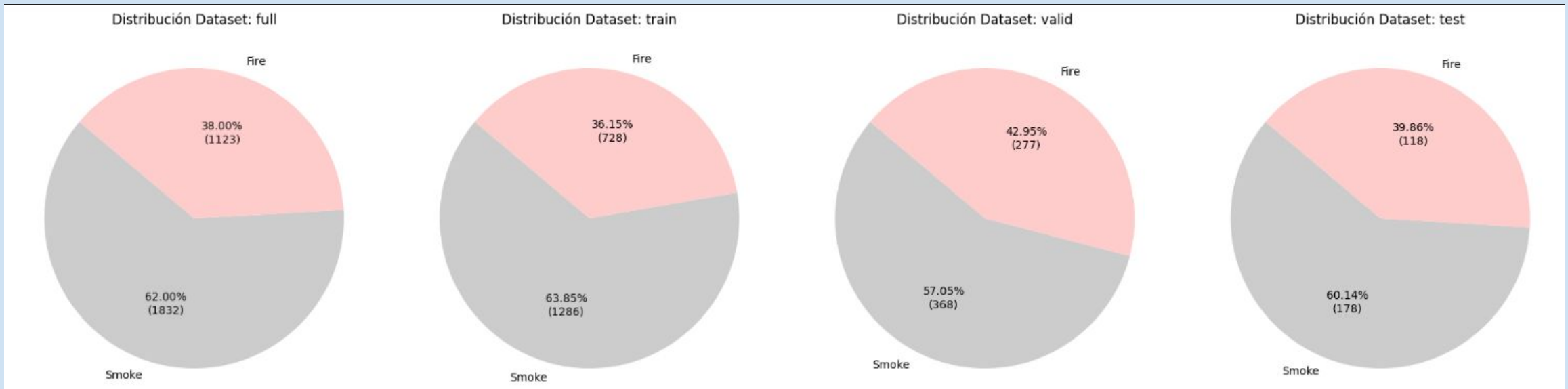
Roboflow:

Balance de Clases:

- Smoke: 1832
- Fire: 1123







Data Augmentation

Para un dataset que tiene como objetivo la detección de fuego y humo, es importante considerar las características visuales de estos fenómenos y cómo pueden variar según las condiciones de iluminación, el entorno, la distancia y las condiciones meteorológicas.

- Transformaciones de color

1 - **Cambio de brillo y contraste:**
El fuego y el humo pueden variar mucho en brillo y contraste dependiendo de la luz ambiental. Si la imagen es tomada durante el día o la noche, o si la imagen tiene una fuente de luz cerca, el brillo y el contraste de la imagen pueden cambiar considerablemente.

2 - **Saturación y matiz:**
El fuego a menudo tiene colores saturados y brillantes (rojos, amarillos, naranjas), mientras que el humo puede variar de gris a blanco. Variar la saturación puede ayudar a que el modelo sea más robusto frente a esas variaciones de color.

3 - **Cambio en la iluminación:**
Dado que el fuego puede ser visible en diferentes condiciones de luz (por ejemplo, a plena luz del día o en la oscuridad), es importante ayudar al modelo a aprender a detectar fuego y humo independientemente de las variaciones en la iluminación ambiental.

Data Augmentation

Transformaciones de Orientación

- **Volteo horizontal aleatorio** para aprender a reconocer patrones de humo y fuego independientemente de su orientación izquierda/derecha. Esto es crucial en imágenes forestales donde la dirección del humo o las llamas no tiene un patrón fijo.
- **Rotación aleatoria de ±30 grados** para mejorar la robustez frente a inclinaciones aleatorias. Dado que las imágenes pueden capturarse desde diferentes ángulos, esta transformación asegura que el modelo detecte correctamente el humo o fuego sin depender de una orientación específica.

Transformaciones de Composición Espacial

- **Recorte y redimensionado aleatorio** que permite al modelo adaptarse a distintas posiciones y escalas del humo y fuego dentro de la imagen. Esto es útil para detectar focos pequeños de fuego o dispersión de humo en áreas extensas.

Conclusión: Aplicar estas augmentaciones aumenta significativamente la capacidad del modelo para generalizar y detectar humo y fuego en condiciones adversas y diversas, haciendo que sea más confiable para su uso en prevención y monitoreo de incendios forestales.

Arquitecturas

Se exploraron y realizaron experimentos con diferentes arquitecturas y configuraciones sobre el conjunto de datos propuesto.

- Modelo SSD ResNet50 V1 FPN
- Retinanet
- Yolo v8

En términos generales cada uno de estos modelos cuentan con diferentes características que a priori nos parecieron adecuadas para iniciar el trabajo.

Arquitecturas

SSD ResNet50 V1 FPN

- Utilidad: Detección rápida de objetos en diferentes tamaños.
- Ventajas: Detección multiescala, velocidad razonable.
- Desventajas: Menor precisión en entornos complejos, no es el más rápido.

Retinanet

- Utilidad: Detección precisa de objetos difíciles.
- Ventajas: Reduce falsos negativos, detecta en múltiples escalas.
- Desventajas: Menor velocidad que YOLOv8, requiere alto poder de cómputo.

YOLO v8

- Utilidad: Detección en tiempo real.
- Ventajas: Alta eficiencia en tiempo real, fácil de adaptar.
- Desventajas: Puede fallar en objetos pequeños, riesgo de falsos positivos.

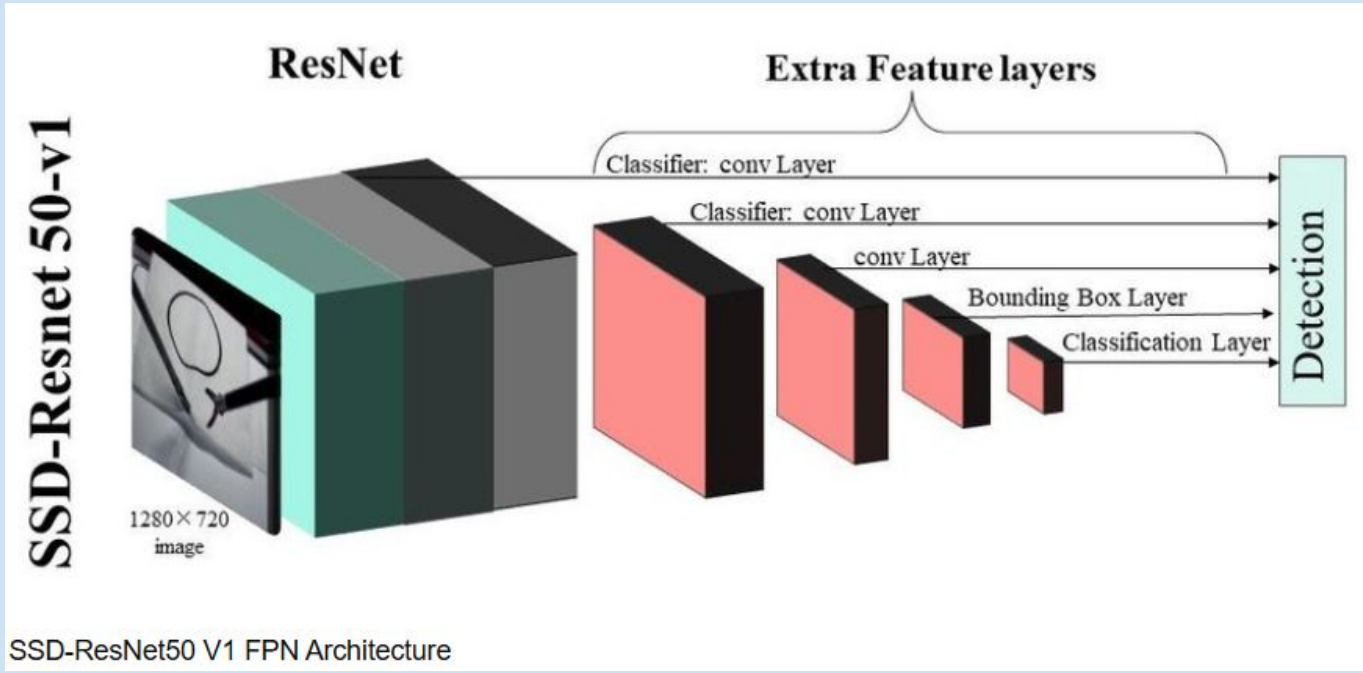
Este resumen se enfoca en comparar utilidad, ventajas y desventajas de cada modelo para la detección de humo y fuego, destacando cuándo sería más conveniente usar cada uno.

Modelo SSD ResNet50 V1 FPN

- SSD (Single Shot MultiBox Detector): Arquitectura de detección de objetos en una sola pasada.

- Utiliza ResNet50 como backbone para la extracción de características y Feature Pyramid Network (FPN) para manejar objetos de diferentes escalas de manera eficiente.

- Utiliza una serie de anchors (anclas) de diferentes tamaños y relaciones de aspecto en cada ubicación de la imagen para predecir la presencia de objetos.



SSD-ResNet50 V1 FPN Architecture

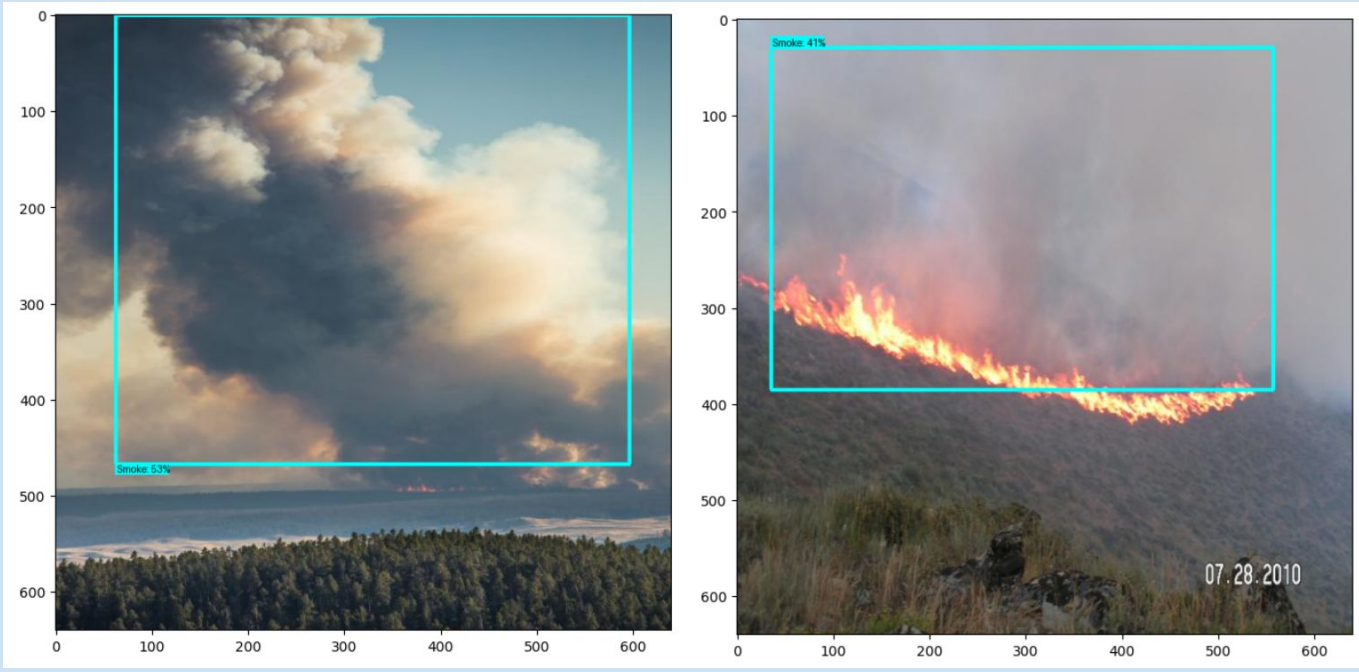
Modelo SSD ResNet50 V1 FPN

- Se entrenó el modelo tal que cuando la loss se estancaba se variaban los pesos de loss de clasificación y BB.

- Se varió el optimizador con Adam y SGD.

- Aunque el modelo llegó a aprender, las métricas finales no fueron satisfactorias.

- Quizás con mejor tunning de hiperparámetros o probando con la Data Augmentation podría haberse obtenido mejores resultados.




```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.001
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.002
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.029
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.049
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.084
```

Modelo Retinanet

- Modelo Base: "COCO-Detection/retinanet_R_50_FPN_1x.yaml"
  - R_50: ResNet-50 como backbone (50 capas).
  - FPN: Feature Pyramid Network (técnica que se utiliza para mejorar la detección de objetos en diversas escalas)
  - 1x: Modelo entrenado con 270,000 iteraciones (la cantidad de veces que el modelo pasa por todo el conjunto de datos durante el entrenamiento).

- MODEL.RETINANET.NUM_CLASSES = 2
- SOLVER.BASE_LR = 0.001
- LR_SCHEDULER_NAME: WarmupMultiStepLR

- 50 épocas con y sin pipeline de augmentations
- 100 épocas con pipeline de augmentations

Modelo Retinanet

FOCAL LOSS

Focal Loss es la principal innovación que RetinaNet introduce para abordar el desequilibrio entre clases.

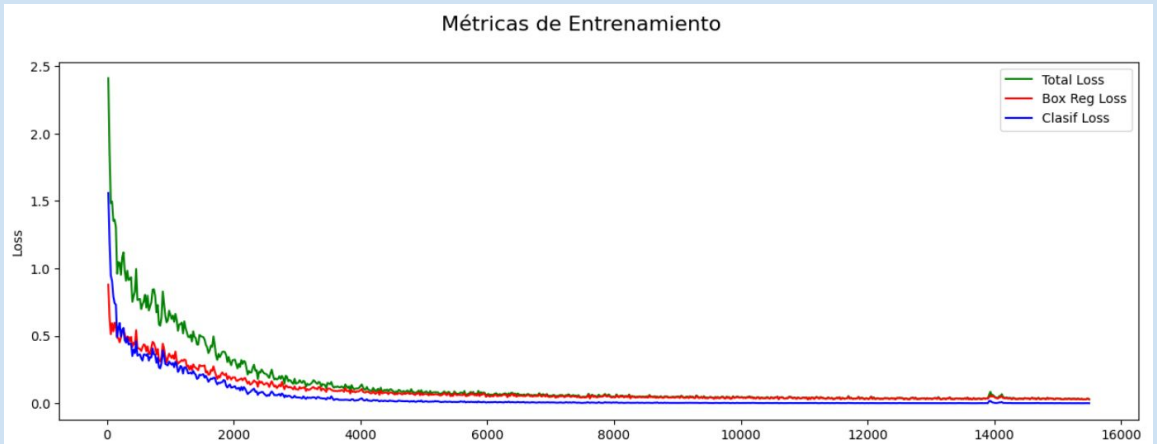
En lugar de usar la tradicional *cross-entropy loss*, que penaliza igualmente las predicciones correctas e incorrectas, Focal Loss reduce la pérdida para las predicciones bien clasificadas y pone más énfasis en las predicciones incorrectas,

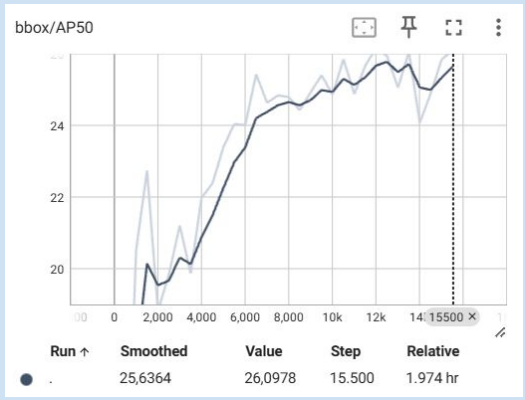
alpha: controla el equilibrio entre las clases positivas (objetos) y las negativas (fondo). (Entre 0 y 1). Un valor más alto de alpha hace que las clases positivas (objetos) sean más relevantes en la pérdida, lo que puede ser útil cuando las clases positivas son menos frecuentes que las negativas.

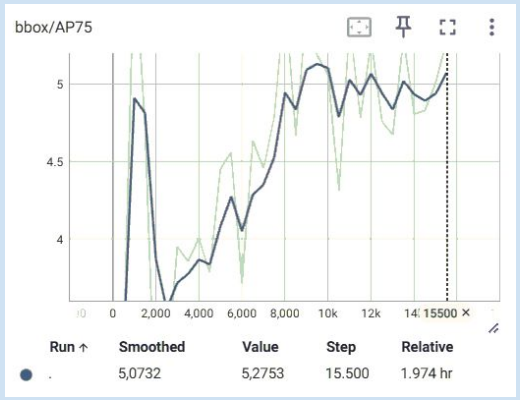
gamma: controla la énfasis que se le da a los ejemplos difíciles (incorrectamente clasificados). (Entre 0 y 5). A medida que gamma aumenta, la función de pérdida se vuelve más focalizada en los ejemplos difíciles y menos en los fáciles.

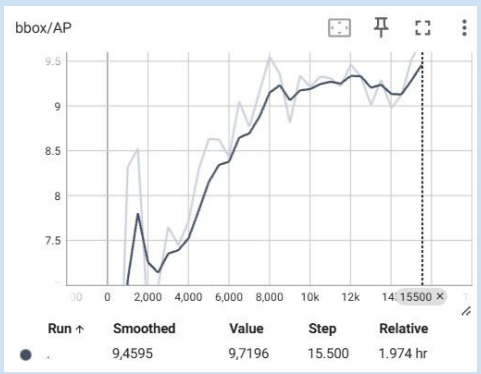
loss_weight: peso general que se asigna a la Focal Loss en el cálculo total de la pérdida. (valor escalar que determine la importancia relativa de la Focal Loss frente a otras pérdidas en el modelo)

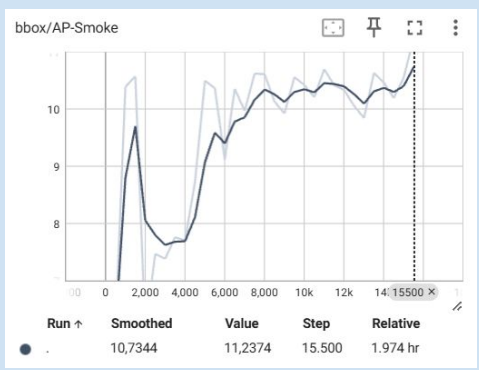
Modelo Retinanet

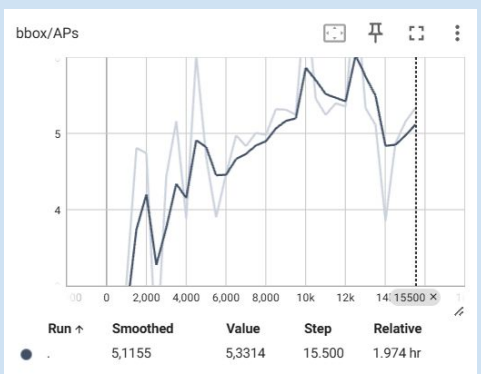


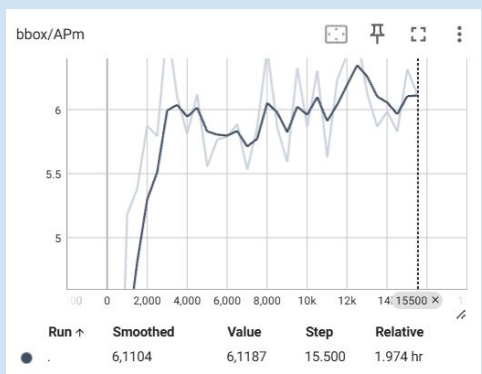


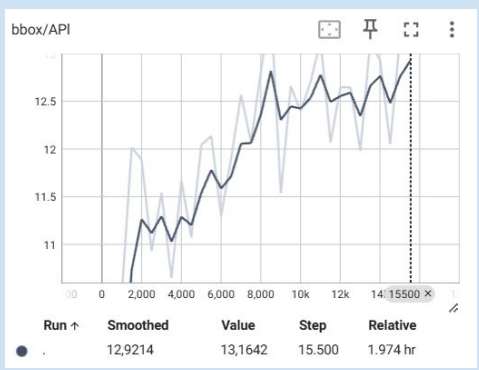












- mAP relativamente bajo (11.6%)
- mAP c/umbral de IoU=0.5 bastante más alto (29.5%) (mejor cuando se permite un margen mayor de error en la coincidencia de las cajas delimitadoras)
- mAP c/ umbral de IoU=0.75 (6.94%) (a umbral más estricto, desempeño menos sólido -> **modelo aún no completamente afinado**)
- Clasifica mejor Humo que Fuego.
- Clasifica mejor objetos grandes.

Modelo Retinanet

- Punto de partida para afinar el modelo:
  - mejor etiquetado de datos
  - mejor balance de clases
  - técnicas de data augmentation
  - cambios en la tasa de aprendizaje
  - cambios en los parámetros de focal loss

Modelo YOLO V8

YOLO v8 es la octava versión de la famosa arquitectura YOLO y sigue mejorando la precisión, velocidad y facilidad de implementación de las versiones anteriores. Esta versión también se centra en mantener un equilibrio entre eficiencia en tiempo de inferencia y precisión de detección.

- Convolutional Backbone: Utiliza una red convolucional para extraer características de la imagen, mejorada para eficiencia y precisión.

- Feature Pyramid Network (FPN): Facilita la detección de objetos de diferentes tamaños mediante el uso de múltiples escalas de características.

- Head de Detección: Realiza predicciones sobre las clases y las ubicaciones de los objetos en tiempo real, dividiendo la imagen en una cuadrícula.

Modelo YOLO V8

Modelo YOLO("yolov8n.pt")

```python
#Entrenamiento sin augmentación
print("Entrenando modelo sin augmentación...")
results_no_aug = model_no_aug.train(
    data="/content/datasets_no_aug/dataset_no_aug.yaml",
    epochs=50,
    batch=32,
    imgsz=640,
    workers=4,
    project="/content/runs",
    name="yolov8_no_aug",
    device=0,
    conf=0.35,       # Ajustar el umbral de confianza
    iou=0.6          # Ajustar el umbral de IoU
)
```

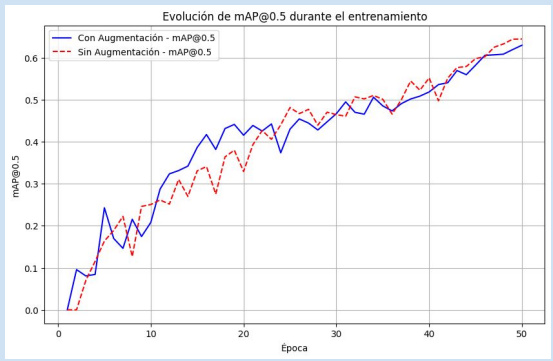
Clase	Imágenes	Instancias	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5-0.95
All	225	733	0.769	0.496	0.643	0.366
Fire	79	279	0.789	0.455	0.632	0.333
Smoke	210	454	0.748	0.537	0.654	0.399

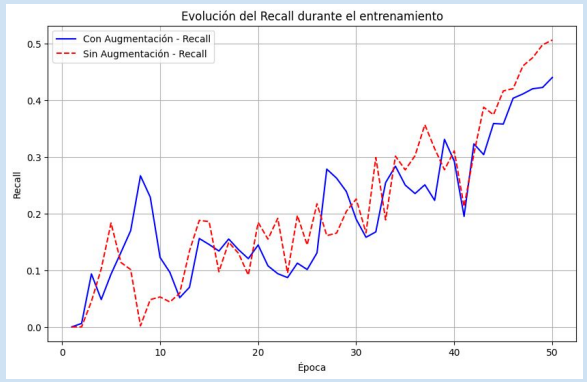
```python
# Entrenamiento con augmentación
print("Entrenando modelo con augmentación...")
results_aug = model_aug.train(
    data="/content/datasets_aug/dataset_aug.yaml",
    epochs=50,
    batch=32,
    imgsz=640,
    workers=4,
    project="/content/runs",
    name="yolov8_aug",
    device=0,
    conf=0.35,       # Ajustar el umbral de confianza
    iou=0.5          # Ajustar el umbral de IoU
)
```

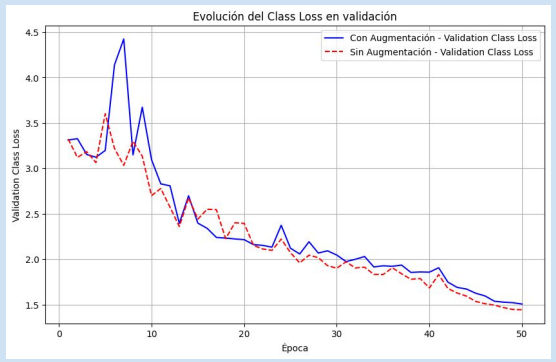
Tabla de Resultados de ejecución con data augmentation:

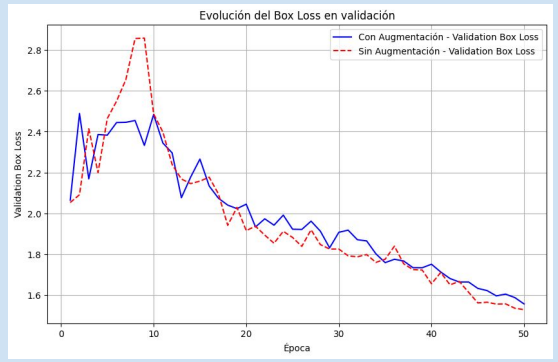
Clase	Imágenes	Instancias	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5-0.95
All	222	721	0.792	0.441	0.629	0.384
Fire	68	234	0.786	0.393	0.601	0.341
Smoke	213	487	0.799	0.489	0.656	0.427

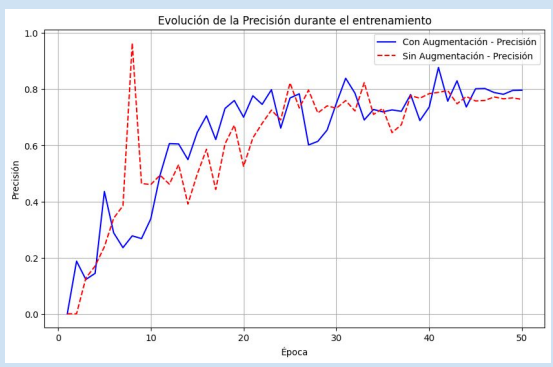
Modelo YOLO V8

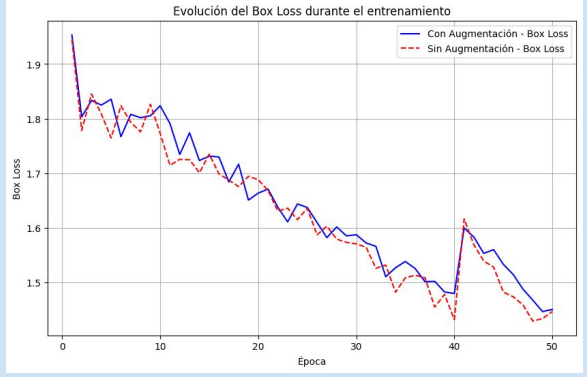


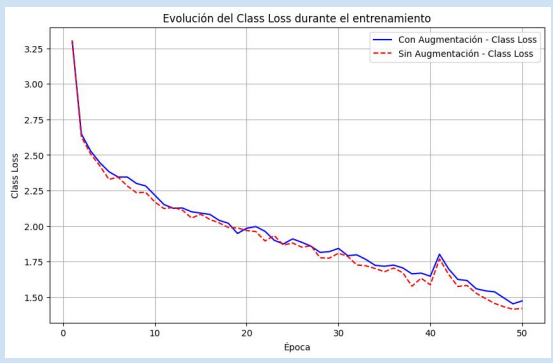












- Precisión (P): El modelo con augmentación avanzada mostró una ligera mejora en la precisión general, especialmente en la clase "smoke", que pasó de 0.748 a 0.799. Esto sugiere que la augmentación ayudó al modelo a aprender más características específicas de los objetos.
- Recall (R): En general, el modelo sin augmentación tuvo un mejor recall (0.496 frente a 0.441). Sin embargo, en el caso específico de la clase "smoke", el recall del modelo con augmentación fue ligeramente más bajo, indicando que, aunque el modelo es más preciso, puede estar perdiendo algunos ejemplos positivos.
- mAP: En cuanto a mAP@0.5-0.95, el modelo con augmentación avanzó de 0.366 a 0.384, y en particular, la clase "smoke" mejoró de 0.399 a 0.427. Esto sugiere una mejora en la capacidad del modelo para detectar objetos más difíciles.

Modelo YOLO V8

Tunning de data augmentation y prueba con YOLO V8m

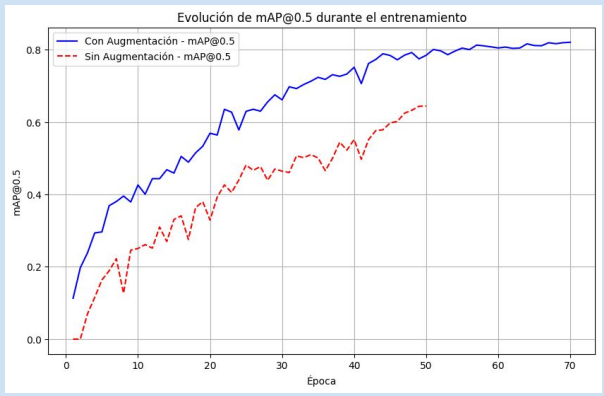
- El cambio del modelo YOLOv8n al modelo YOLOv8m se realizó para aprovechar la mayor capacidad del modelo YOLOv8m en comparación con YOLOv8n.
- YOLOv8m tiene una arquitectura más compleja, con más parámetros y capas, lo cual le permite aprender características más ricas y complejas de los datos..

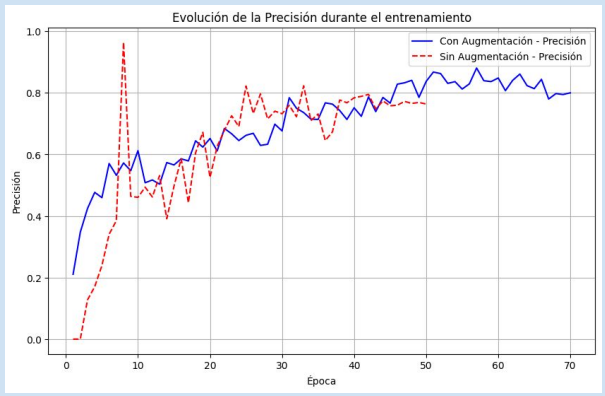
Se realizaron ajustes de hiperparámetros y se obtuvo el siguiente resultado

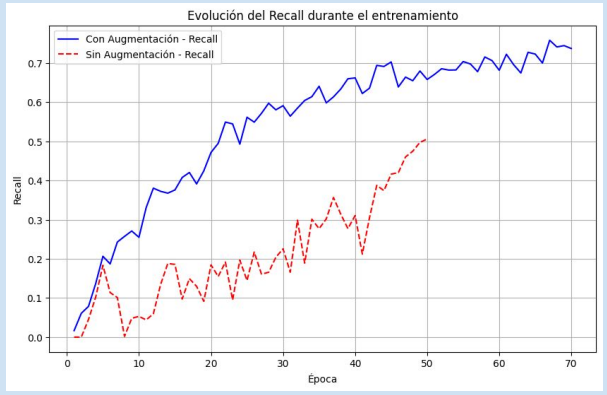
Tabla de Resultados de ejecución con data augmentation yolov8m:

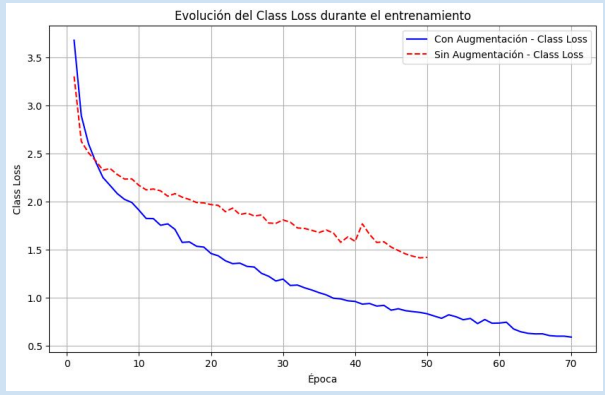
Clase	Imágenes	Instancias	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5-0.95
All	222	721	0.799	0.737	0.821	0.598
Fire	68	234	0.782	0.684	0.780	0.489
Smoke	213	487	0.815	0.791	0.861	0.707

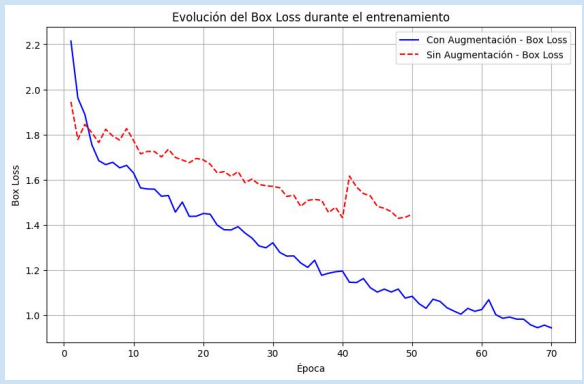
Modelo YOLO V8

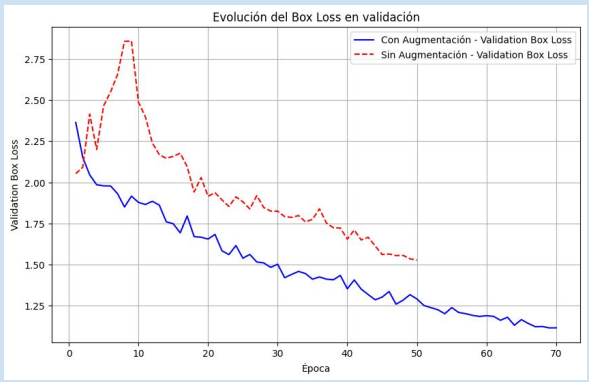










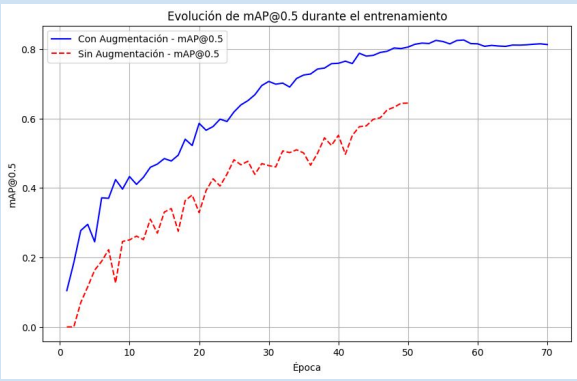


- **Mejor Precisión y Recall**: Con el uso del modelo YOLOv8m y técnicas avanzadas de augmentación, se lograron mejoras notables en las métricas clave como la precisión (P) y el recall (R). La clase "smoke" mostró resultados especialmente sólidos, con un recall del 79.1%, lo cual sugiere que el modelo tiene buena capacidad para detectar humo en diferentes condiciones.

- **Mayor mAP@0.5 y mAP@0.5-0.95**: En comparación con los modelos anteriores (YOLOv8 sin augmentación y YOLOv8 con augmentación), el uso de YOLOv8m resultó en un aumento considerable de mAP@0.5 (82.1%) y mAP@0.5-0.95 (59.8%). Esto indica que el modelo mejora su capacidad general de detección y precisión en la localización de los objetos.

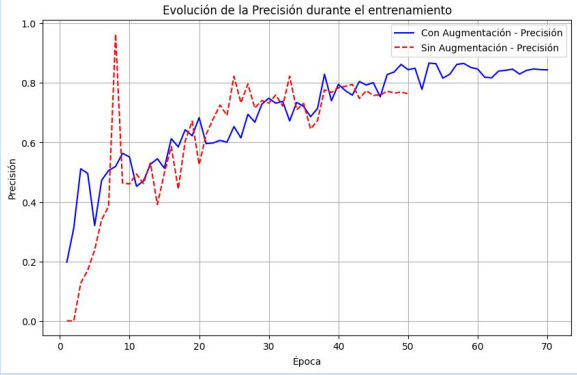
Modelo YOLO V8

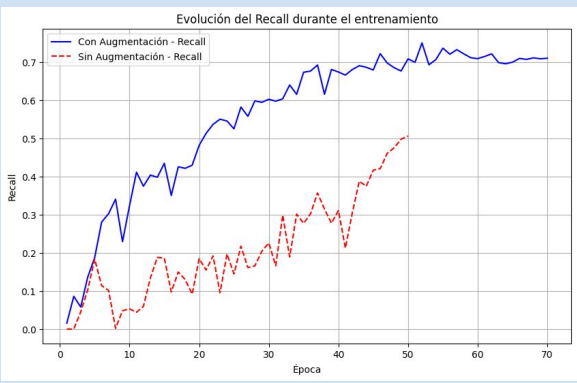
Clase	Imágenes	Instancias	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5-0.95
All	222	721	0.838	0.712	0.813	0.594
Fire	68	234	0.829	0.644	0.767	0.488
Smoke	213	487	0.846	0.781	0.860	0.699



Nuevo tunning aplicado a YOLOv8m

- El modelo alcanzó una precisión global de 83.8% y un recall de 71.2%, mientras que el mAP@0.5-0.95 fue de 59.4%. La clase "smoke" mostró un rendimiento destacado con una precisión de 84.6% y un recall de 78.1%, lo cual refleja que el modelo tiene una mejor capacidad de detección del humo en comparación con el fuego.
- La aplicación de la augmentación y el ajuste de hiperparámetros permitieron mejorar la capacidad de generalización del modelo, mostrando resultados más robustos y consistentes. Esto sugiere que YOLOv8m con un entrenamiento optimizado es una alternativa prometedora para la detección de humo y fuego en imágenes forestales, contribuyendo potencialmente a la prevención de incendios.





Resumen final:

YOLO:

Clase	Imágenes	Instancias	Precisión (P)	Recall (R)	mAP@0.5	mAP@0.5-0.95
All	222	721	0.838	0.712	0.813	0.594
Fire	68	234	0.829	0.644	0.767	0.488
Smoke	213	487	0.846	0.781	0.860	0.699

RetinaNet:

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.097
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.261
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.053
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.053
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.061
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.132
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.102
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.185
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.187
```

El problema plantea que es preferible identificar falsos positivos en lugar de no detectarlos, ya que un aviso erróneo de un incendio resulta menos perjudicial que omitir su detección.

Por esta razón, se prioriza optimizar el Recall, considerando que esta métrica refleja la capacidad del modelo para identificar correctamente todos los casos relevantes.

Bajo este criterio, y nuestros experimentos el modelo que mejor se adapta a las necesidades es YOLO

**Aún queda mucho por profundizar, probar modelos, tunning de hiperparámetros..*

¡Muchas gracias!