FIRE DETECTOR

Alejandro Fernández
&
Andrés García

ÍNDICE

INTRODUCCIÓN

NUESTRO PROYECTO ‘FIRE DETECTOR’ SE BASA EN UNA APLICACIÓN WEB QUE DETECTA FUEGO EN TIEMPO REAL A TRAVÉS DE UNA CÁMARA. ADEMÁS, ES CAPAZ DE ENVIAR UN CORREO ELECTRÓNICO Y EJECUTAR UNA ALARMA SONORA AL DETECTAR FUEGO AL REALIZAR UNA IMAGEN.

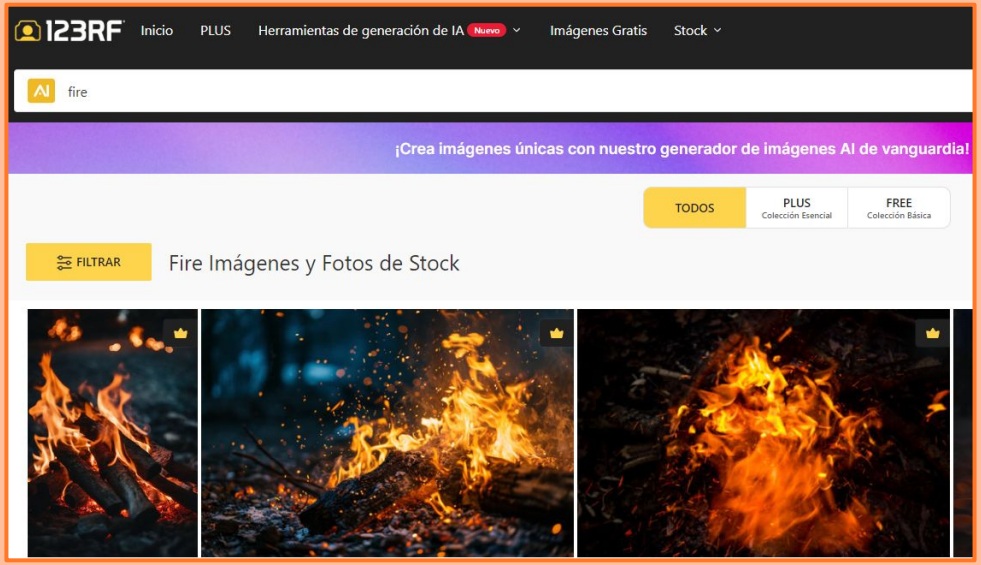
UTILIZA UN ALGORITMO LLAMADO YOLO, ESTE MODELO SE CENTRA EN EL RECONOCIMIENTO DE OBJETOS EN IMÁGENES MEDIANTE LA SEGMENTACIÓN. AL ESTAR ESPECIALIZADO EN ESTE CAMPO, HACE QUE LOS MODELOS SEAN MUCHO MÁS VELOCES Y PRECISOS.

OBTENCIÓN DE DATOS

AL UTILIZAR EL ALGORITMO YOLOV8, NECESITAMOS POR UN LADO LAS IMÁGENES PARA ENTRENAR EL MODELO Y POR OTRO LA SEGMENTACIÓN DE ESTAS IMÁGENES INDICANDO CUÁL ES EL TARGET, EN ESTE CASO, EL FUEGO. PARA ELLO, UNA DE LAS OPCIONES ES UTILIZAR UN DATASET OBTENIDO DESDE LA PÁGINA ROBOFLOW.

OBTENCIÓN DE DATOS

LA OTRA OPCIÓN ELEGIDA ES OBTENER IMÁGENES MEDIANTE WEB SCRAPING, EN ESTE CASO A LA PÁGINA 123RF.

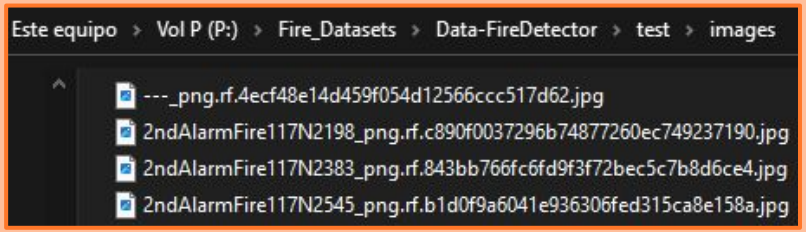


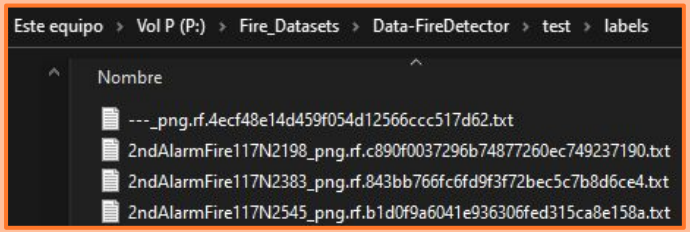

```python
from google.colab import files
# Descargar y guardar cada imagen
for i, img in enumerate(imagenes_fuego, start=1):
    # Obtener la URL de la imagen
    url_imagen = urljoin(url_base, img['src'])

    # Descargar la imagen
    respuesta_imagen = requests.get(url_imagen)

    # Guardar la imagen en el directorio
    nombre_archivo = os.path.join(directorio_imagenes, f'imagen_{i}.png')
    with open(nombre_archivo, 'wb') as archivo:
        archivo.write(respuesta_imagen.content)

    print(f'Imagen {i} descargada y guardada como {nombre_archivo}')

# Comprimir el directorio en un archivo ZIP
nombre_archivo_zip = 'imagenes_fuego.zip'
os.system(f'zip -r {nombre_archivo_zip} {directorio_imagenes}')

# Descargar el archivo ZIP
files.download(nombre_archivo_zip)
```

LIMPIEZA

EL DATASET UTILIZADO ESTÁ ESTRUCTURADO EN ÁRBOLES DE CARPETAS.

- ENCONTRAMOS TRES RAMAS IMPORTANTES: TRAIN, TEST Y VALID. EN CADA UNA DE ESTAS CARPETAS HAY UNA CARPETA QUE GUARDA LAS IMÁGENES Y OTRA QUE CONTIENE LOS ARCHIVOS .TXT PARA INDICAR LA SEGMENTACIÓN DE LA IMAGEN.





SI NOS FIJAMOS EN LA CARPETA TRAIN, NOS DAMOS CUENTA LAS IMÁGENES TIENEN LA MISMA ESTRUCTURA: [NOMBRE]_PNG.RF.[IDENTIFICADOR].JPG. ADEMÁS, ESTE IDENTIFICADOR LO CONTIENE EL MISMO ARCHIVO .TXT ASOCIADO A ESTA IMAGEN EN SU RESPECTIVA CARPETA.

LIMPIEZA

ESTA INFORMACIÓN LA USAREMOS PARA COMPROBAR QUE CADA IMAGEN TENGA UN LABEL ASOCIADO, ASÍ COMO QUE CADA CARPETA CONTIENE ARCHIVOS DE EXTENSIÓN CORRECTA (.TXT PARA LOS LABELS Y .PNG O .JPG PARA LAS IMÁGENES).

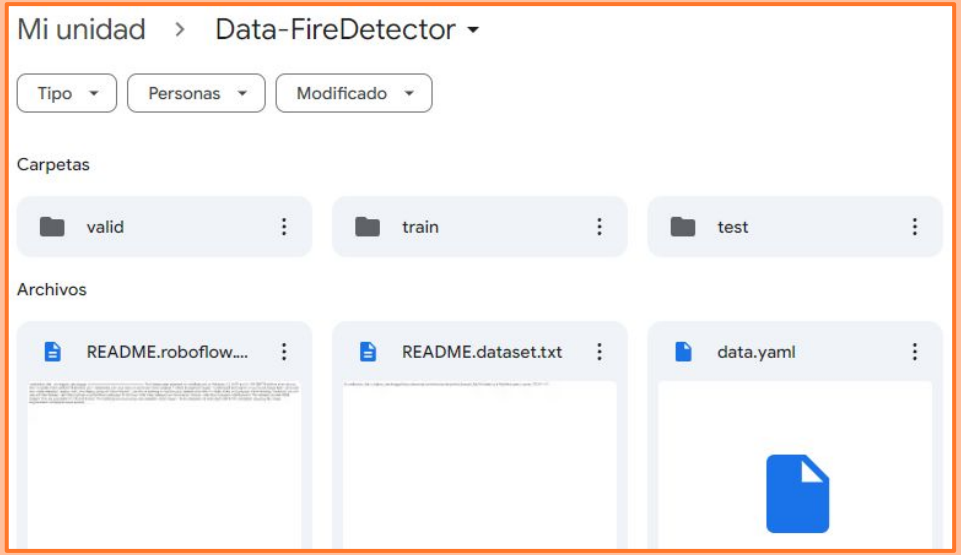
SI NO OCURRE ESTO, SE GUARDARÁ ESTE ARCHIVO O CARPETA EN UNA NUEVA CARPETA CREADA PARA DECIDIR QUÉ HACER CON ELLA.

```
    # Comprobamos por cada imagen si tiene un label asociado
    for archivo_image in archivos_images:

        # Verificamos que el archivo sea .jpg o .png . Si es así guardamos su identificador
        if archivo_image.endswith(".jpg") or archivo_image.endswith(".png"):
            identificador = archivo_image.split(".")[-2]
        else:
            ruta_imagen = os.path.join(carpeta_images, archivo_image)
            ruta_destino = f"/content/continuous_fire-6/No_Compatibles/{nombre}/"

            print(f"""El archivo {archivo_image}, en la carpeta {nombre} no es un archivo .jpg o .png.\n
Será enviado a la carpeta {ruta_destino} \n""")

            try:
              os.mkdir(ruta_destino)
              shutil.move(ruta_imagen, ruta_destino)

            except:
              shutil.move(ruta_imagen, ruta_destino)

        # Verificar si existe un archivo con el mismo identificador en la carpeta "labels"
        if identificador in identificadores_labels:
            cont[nombre] += 1
        else:
            print(f"""La imagen con identificador {identificador} no tiene un archivo en la carpeta 'labels' en {nombre}.
Eliminando la imagen... \n""")
            ruta_imagen = os.path.join(carpeta_images, archivo_image)
            os.remove(ruta_imagen)

for nombre in carpetas:
  print(f"Número de imágenes en la carpeta {nombre:5} con label asociado: {cont[nombre]}")
```

EXPLORACIÓN Y VISUALIZACIÓN

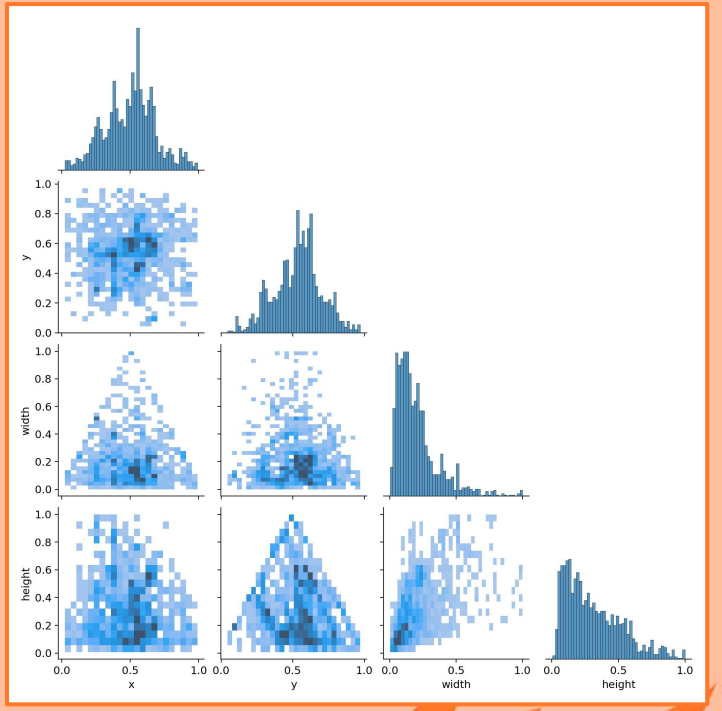
EN ESTE EJEMPLO, COPIAMOS EL DATASET A NUESTRO GOOGLE DRIVE PARA EXPLORAR LOS ARCHIVOS QUE CONTIENE:

- ARCHIVOS README SOBRE EL ORIGEN DE LOS DATOS.
- ARCHIVO DATA.YAML QUE SIRVE PARA CONFIGURAR LAS RUTAS DE LAS IMÁGENES PARA EL ENTRENAMIENTO.
- TRES CARPETAS: VALID, TRAIN Y TEST QUE CONTIENEN LAS IMÁGENES JUNTO CON SUS RESPECTIVAS SEGMENTACIONES.



EXPLORACIÓN Y VISUALIZACIÓN

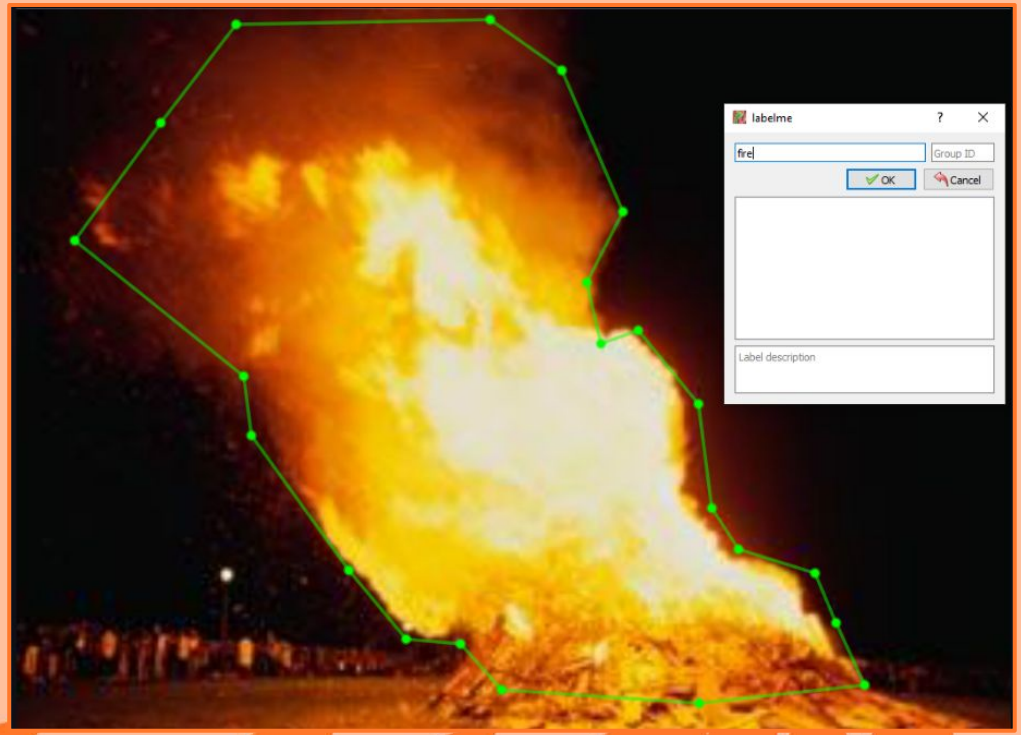
UNA VEZ SE ENTRENA EL MODELO, YOLO NOS GENERA VARIOS TIPOS DE GRÁFICAS COMO LA MATRIZ DE CORRELACIÓN ENTRE LA **ANCHURA** Y **ALTURA** DE LA **IMAGEN** Y LA **ZONA DETECTADA** CON LAS COORDENADAS (X, Y) DE CADA SEGMENTACIÓN.



PREPARACIÓN DE LOS DATOS PARA LOS ALGORITMOS DE MACHINE LEARNING

CON LABELME SEGMENTAMOS LAS IMÁGENES.

UNA VEZ SEGMENTADA UNA IMAGEN, SE GENERA UN ARCHIVO .JSON QUE INDICA LAS COORDENADAS DE LA SEGMENTACIÓN DEL TARGET EN LA IMAGEN.



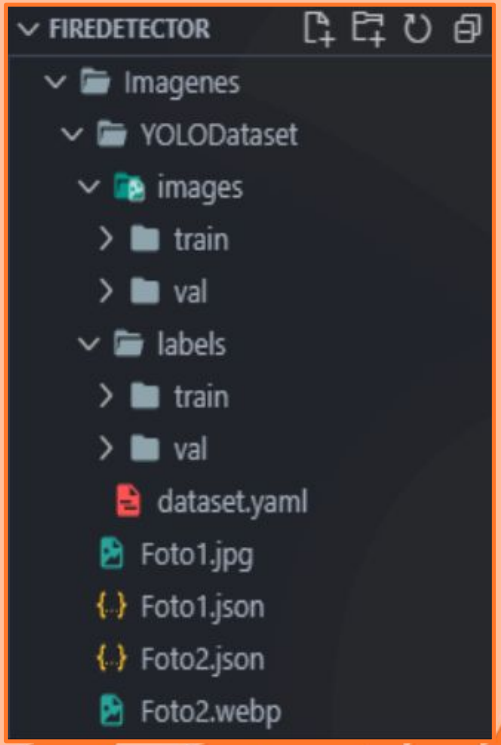
PREPARACIÓN DE LOS DATOS PARA LOS ALGORITMOS DE MACHINE LEARNING

CON **LABELME2YOLO** TRANSFORMAMOS LOS DATOS PARA EL ALGORITMO.

AL EJECUTAR LA TRANSFORMACIÓN CON:

LABELME2YOLO --JSON_DIR "RUTA DE LA CARPETA CON LAS IMÁGENES"

SE GENERA UNA CARPETA QUE SEPARA POR UN LADO LAS IMÁGENES DE LOS ARCHIVOS .JSON ASOCIADOS, Y CREA UN ARCHIVO **DATASET.YAML** QUE SERÁ EL QUE UTILIZAREMOS PARA REALIZAR EL ENTRENAMIENTO.



ENTRENAMIENTO Y COMPROBACIÓN

MODELO YOLOV8

ROBOFLOW: PARA DESCARGAR EL DATASET DE IMÁGENES

```
rf = Roboflow(api_key="MAiCeSuy58yjlg2ma4QK")
project = rf.workspace("-jwzpw").project("continuous_fire")
dataset = project.version(6).download("yolov8")
```

ULTRALYTICS: PARA OBTENER Y ENTRENAR EL MODELO.

```
!yolo task=detect mode=train model=yolov8s.pt data=/content/continuous_fire-6/data.yaml epochs=80 imgsz=640 plots=True
```

ENTRENAMIENTO Y COMPROBACIÓN

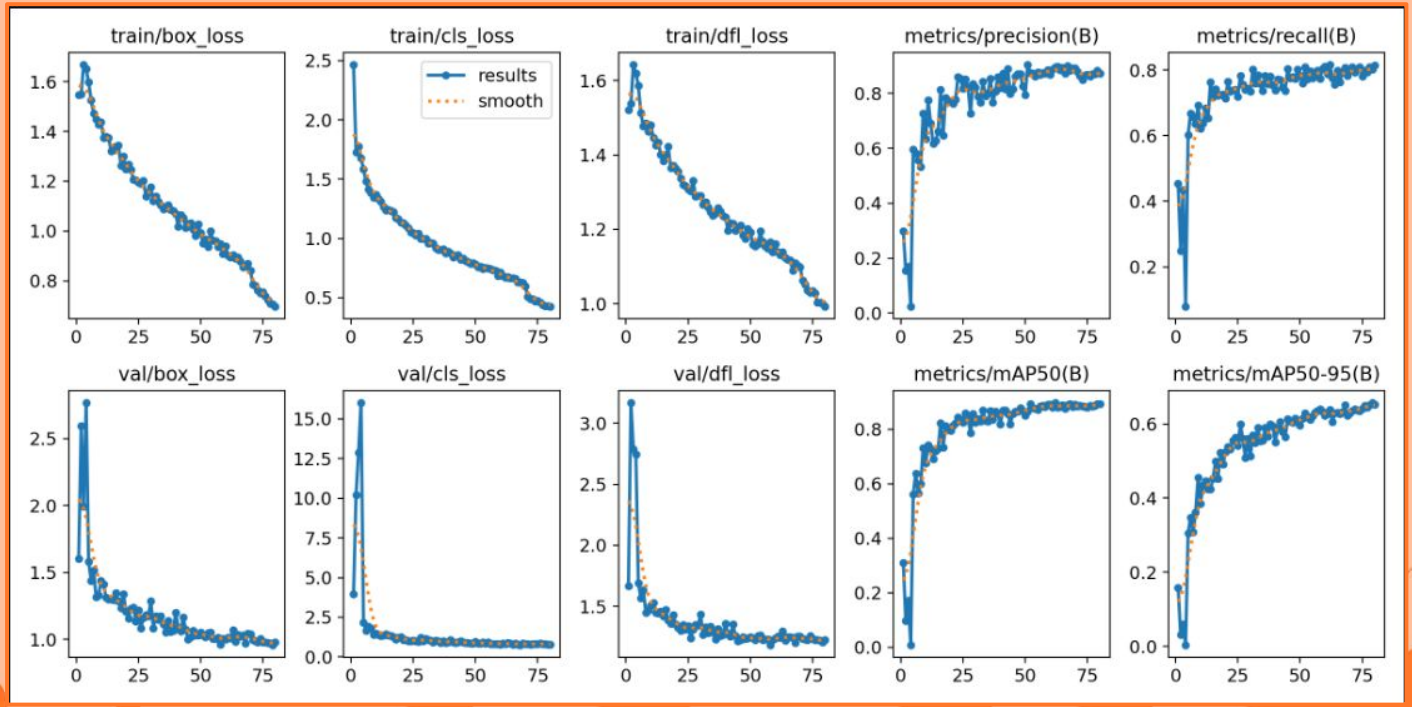
MODELO YOLOV8

TRAS TERMINAR EL ENTRENAMIENTO VEMOS UN RESUMEN DE LAS ITERACIONES QUE HA REALIZADO. NOS FIJAMOS QUE EL MODELO HA ALCANZADO UNA PRECISIÓN DEL **88,31%**.

epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)
1	1.5482	2.4676	1.52	0.2994	0.45368	0.31052
2	1.5498	1.7261	1.5374	0.15422	0.24947	0.09754
3	1.6696	1.7763	1.6429	0.17021	0.43368	0.1711
4	1.6521	1.678	1.6184	0.02385	0.07895	0.00919
5	1.5977	1.5831	1.5863	0.59603	0.60211	0.56022
...	...	...	...	...	...	...
75	0.75444	0.47607	1.0348	0.86927	0.78105	0.87979
76	0.73771	0.46127	1.0283	0.86209	0.79789	0.88628
77	0.72293	0.44233	1.0044	0.87006	0.79368	0.88495
78	0.70837	0.43255	1.0045	0.87084	0.80947	0.88686
79	0.70564	0.4307	1.0002	0.8831	0.80211	0.89361
80	0.69584	0.42559	0.99257	0.87153	0.81407	0.89305

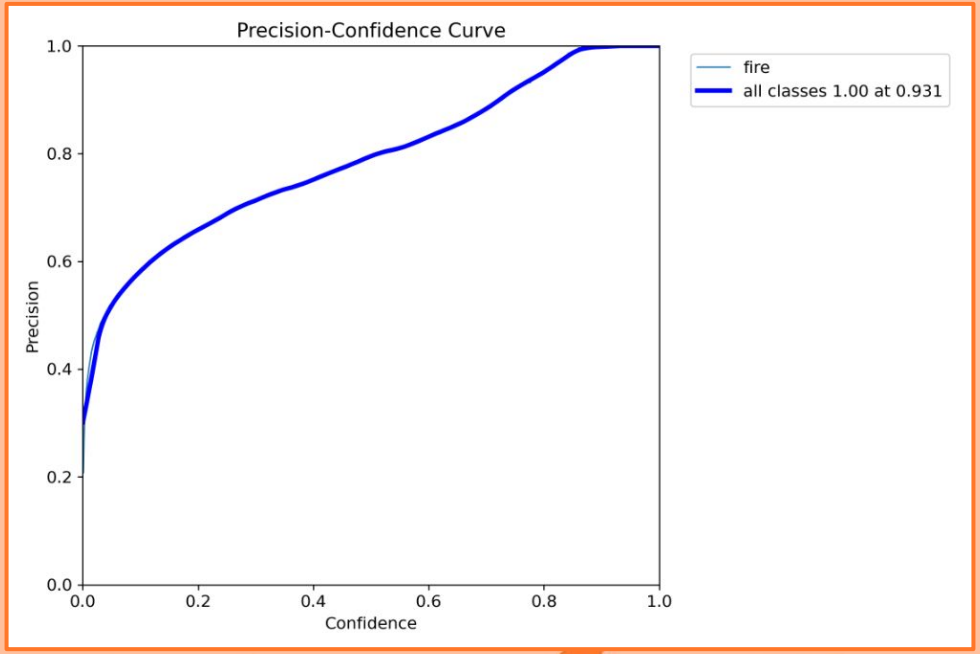
ENTRENAMIENTO Y COMPROBACIÓN

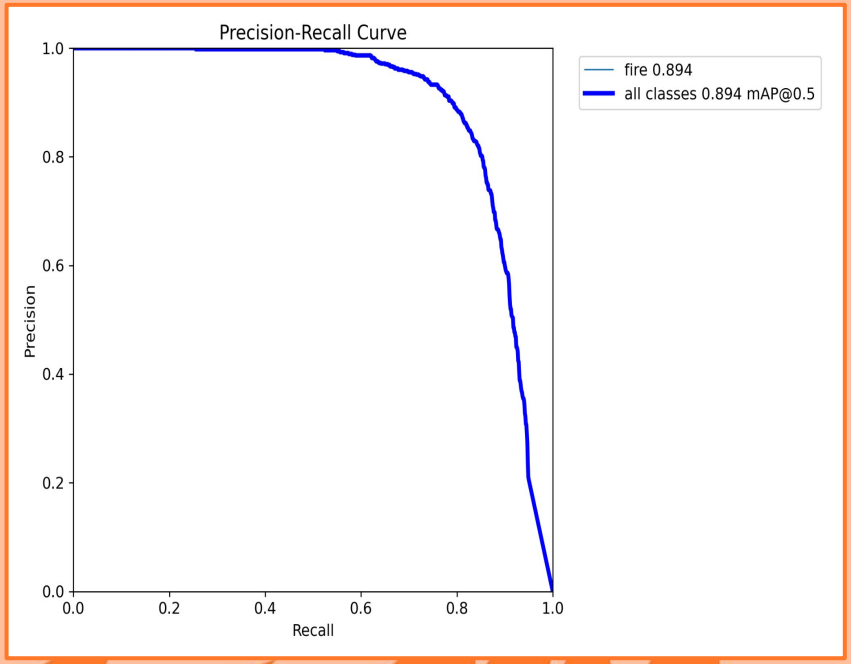
MODELO YOLOV8



ENTRENAMIENTO Y COMPROBACIÓN

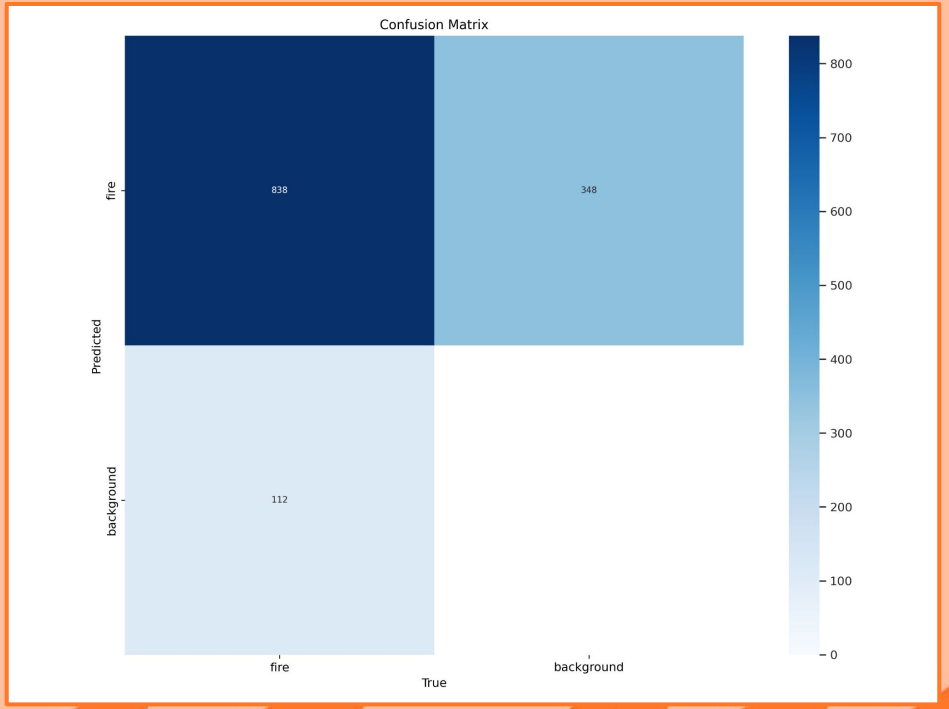
MODELO YOLOV8





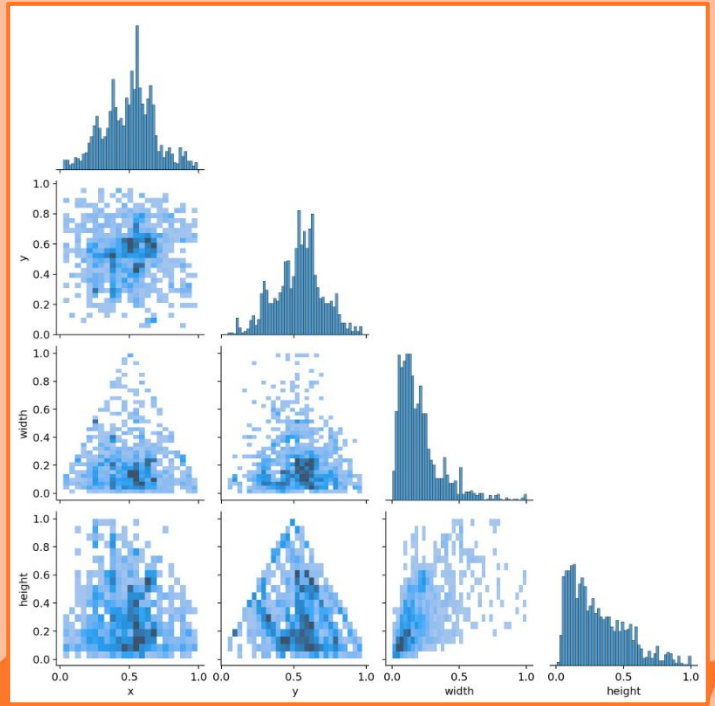
ENTRENAMIENTO Y COMPROBACIÓN

MODELO YOLOV8



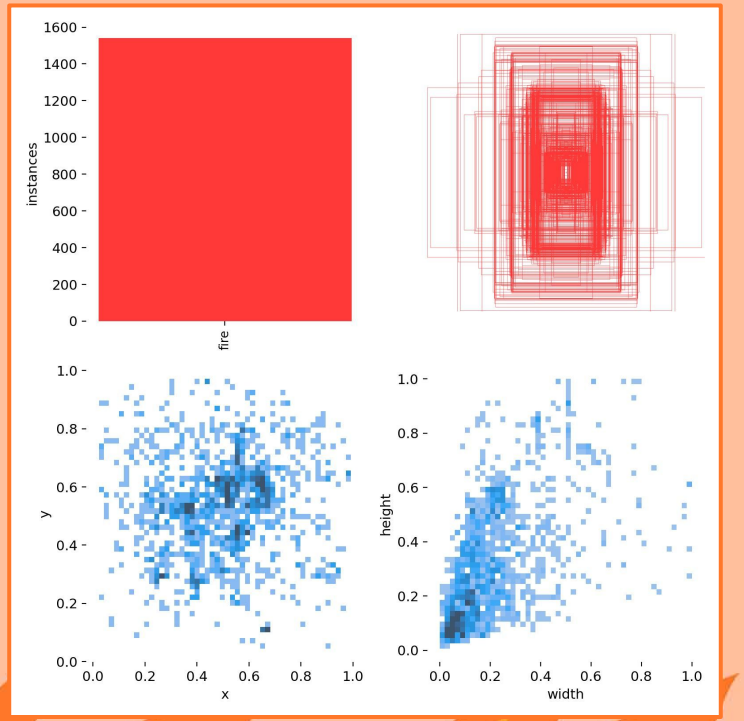
ENTRENAMIENTO Y COMPROBACIÓN

MODELO YOLOV8



ENTRENAMIENTO Y COMPROBACIÓN

MODELO YOLOV8



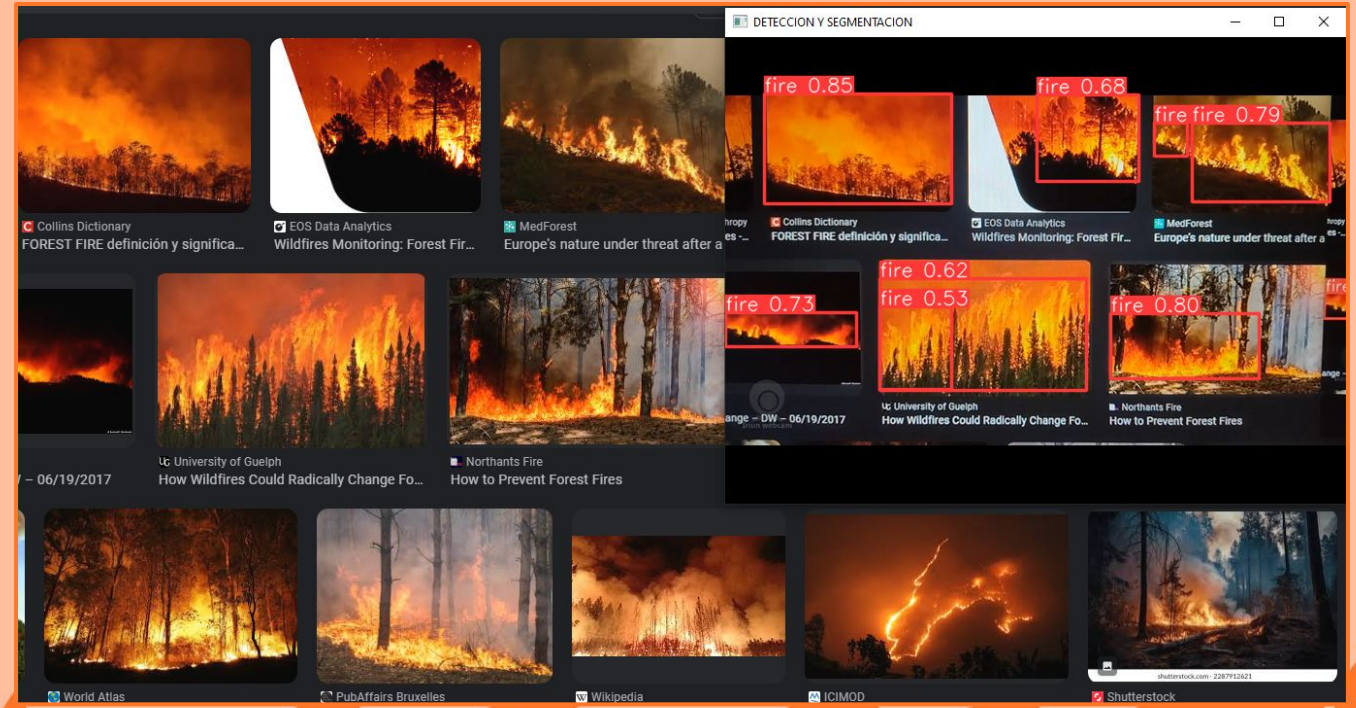
ENTRENAMIENTO Y COMPROBACIÓN

RESULTADOS CON CÁMARA

```python
# Leer nuestro modelo, tenemos que especificar la ruta del modelo.
model = YOLO("best.pt")
# Capturar video
cap =  cv2.VideoCapture(1)

# Bucle
while True:
    # Leer fotogramas
    ret, frame = cap.read()

    # Leemos resultados
    resultados = model.predict(frame, imgsz = 640, conf=0.80)

    # Mostramos resultados
    anotaciones = resultados[0].plot()

    # Mostramos nuestros fotogramas
    cv2.imshow("DETECCION Y SEGMENTACION", anotaciones)

    # Cerrar nuestro programa
    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

ENTRENAMIENTO Y COMPROBACIÓN

RESULTADOS CON CÁMARA



NPL

1. RECONOCIMIENTO DE VOZ

HEMOS CREADO UN BOTÓN EN LA PÁGINA WEB QUE DETECTA EL MICRÓFONO DEL USUARIO.

UNA VEZ RECONOCE LA VOZ, SE REALIZA EL PROCESAMIENTO DEL LENGUAJE NECESARIO PARA:

- CONVERTIR EL TEXTO A MINÚSCULAS.
- ELIMINAR LAS TILDES.
- TOKENIZAR EL TEXTO.
- ELIMINAR LAS STOP WORDS.


```python
def procesar_texto(texto):
    # Convertir a minúsculas
    texto = texto.lower()

    # Eliminar tildes
    texto = unidecode.unidecode(texto)

    # Tokenización
    tokens = word_tokenize(texto)

    # Eliminar stopwords
    stop_words = set(stopwords.words('spanish'))
    tokens = [word for word in tokens if word not in stop_words]

    # Comprobamos si alguno de los tokens coincide con uno de los apartados
    for token in tokens:
        if token in pre_titles:
            # Si uno de los tokens coincide devolvemos esta palabra
            return token

    # Unir tokens en un solo string
    processed_text = ' '.join(tokens)

    # Si ningun token coincide devolvemos todo el texto procesado
    return processed_text
```

NPL

2. ALARMA SONORA

CUANDO AL TOMAR UNA FOTO SE DETECTE FUEGO, SONARÁ AUTOMÁTICAMENTE UNA VOZ AVISANDO AL USUARIO.

```python
# Pasar de texto a audio
# Parametros:
# - text: Texto para pasar a audio
# - lang: Idioma con el que convertir el texto
# - slow: Indicar la velocidad de reproduccion del audio
def text_to_speech(text, lang='en', slow=False):
    tts = gTTS(text=text, lang=lang, slow=slow)
    return tts
```

HEMOS INTRODUCIDO LA POSIBILIDAD PARA QUE NOS AVISE EN DIFERENTES IDIOMAS:

```python
LANGUAGES = {
    'English': 'en',
    'Spanish': 'es',
    'French': 'fr',
    'German': 'de',
    'Italian': 'it',
}
```

```python
texto_audio = {"es": "ALERTA! FUEGO DETECTADO!",
               "en": "Alert, fire detected",
               "fr": "Alerte, incendie détecté",
               "de": "Alarm, Feuer erkannt",
               "it": "Allerta, rilevato incendio"}
```

NPL

3. VERIFICACIÓN DE CORREO ELECTRÓNICO

AL AÑADIR LA POSIBILIDAD AL USUARIO DE INTRODUCIR UN CORREO ELECTRÓNICO PARA AVISARLE AL DETECTAR FUEGO, TENEMOS QUE ASEGURARNOS DE QUE ESTE CORREO ES CORRECTO.

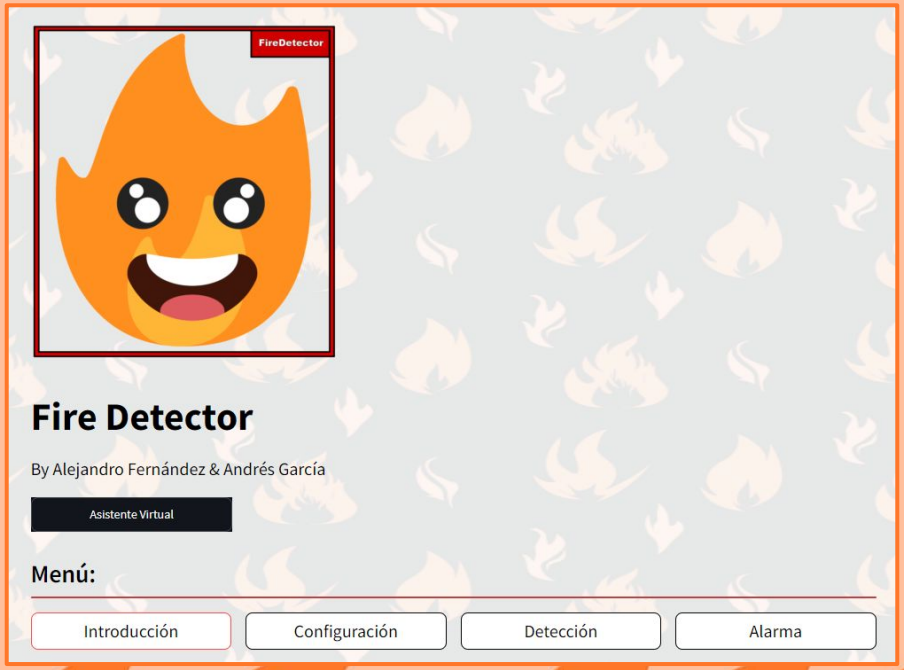
PARA CONSEGUIRLO, USAMOS LA SIGUIENTE EXPRESIÓN REGULAR:

```python
def validar_correo(email):
    # Expresión regular para validar correo electrónico
    regex = r'^(([^<>()\[\]\\.,;:\s@\"]+(\.[^<>()\[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$'

    # Validar el correo con la expresión regular
    if re.match(regex, email):
        return True
    else:
        return False
```

APLICACIÓN WEB

PARA VISUALIZAR NUESTRA APLICACIÓN WEB, SE PUEDE ACCEDER DESDE EL SIGUIENTE ENLACE:

HTTPS://FIREDETECTOR-TFM.STREAMLIT.APP



CONCLUSIONES

LA EXPERIENCIA SE ADQUIERE TRABAJANDO CON TECNOLOGÍAS DESCONOCIDAS, ES INMENSA, NO SOLO POR EL TIEMPO QUE SE EMPLEA BUSCANDO INFORMACIÓN, SINO POR LA CANTIDAD DE ERRORES Y PROBLEMAS QUE HAY QUE RESOLVER. PARA NOSOTROS, HERRAMIENTAS COMO YOLO O STREAMLIT HAN SIDO TANTO UNA MARAVILLA COMO UN RETO, PERO ESTAMOS ORGULLOSOS DEL RESULTADO OBTENIDO, DADO QUE HA CUMPLIDO CON CRECES NUESTRAS EXPECTATIVAS.

HEMOS LOGRADO UN BUEN ENTRENAMIENTO QUE LE DA ROBUSTEZ A NUESTRO MODELO, ASÍ COMO UNA PÁGINA CON COMPLEMENTOS Y BIEN DECORADA DENTRO DE LO QUE PERMITE STREAMLIT. EL HECHO DE TRABAJAR CON RECONOCIMIENTO DE IMÁGENES NOS HA PERMITIDO CUBRIR POSIBLES VACÍOS DE CONOCIMIENTO Y NOS HA PREPARADO PARA LAS PRÁCTICAS Y LO QUE VENGA MÁS ADELANTE.

PARA TERMINAR, NOS GUSTARÍA AGRADECER AL PROFESORADO Y AL EXCELENTE CONTENIDO QUE NOS HA OFRECIDO, DEL CUAL PUEDEN SURGIR PROYECTOS Y RELACIONES PERSONALES COMO ESTA.

ATENTAMENTE - ANDRÉS GARCÍA Y ALEJANDRO FERNÁNDEZ

BIBLIOGRAFÍA.

VÍDEO - PRESENTACIÓN TFM: https://drive.google.com/file/d/1jzdzccb6rihnmdo30gsmyfodpp_pnuyl/view?usp=sharing

WEB STREAMLIT - FIREDETECTOR: https://firedetector-tfm.streamlit.app

GITHUB - ULTRALYTICS: https://github.com/ultralytics/ultralytics

WEB DE IMÁGENES - 123RF: https://es.123rf.com/

WEB IMÁGENES - ROBOFLOW: https://universe.roboflow.com/-jwzpw/continuous_fire

COLAB - SCRAPING WEB: https://colab.research.google.com/drive/1mlyd4hseiystt9nfaydvwlkac5oancq2?usp=sharing

COLAB - LIMPIEZA DE DATOS: https://colab.research.google.com/drive/18fpswtwqpxltktj2ofpwert8xqzyq3td?usp=sharing

COLAB - ENTRENAMIENNTO: https://colab.research.google.com/drive/1mmfqi4k9ic9whai8tfcmuotnll3uum4s?usp=sharing

TUTORIAL - MANDAR CORREO DESDE PYTHON: https://www.youtube.com/watch?v=opao8hh8bjo

TUTORIAL - ENTRENAR MODELO YOLO: https://www.youtube.com/watch?v=rk?zobrjwcc

TUTORIAL - WEBCAM STREAMLIT: https://medium.com/mlearning-ai/live-webcam-with-streamlit-f32bf68945a4