TFG - VIU - Carlos Romero Matarin

https://code.earthengine.google.com/

1. Carga de librerías y archivos

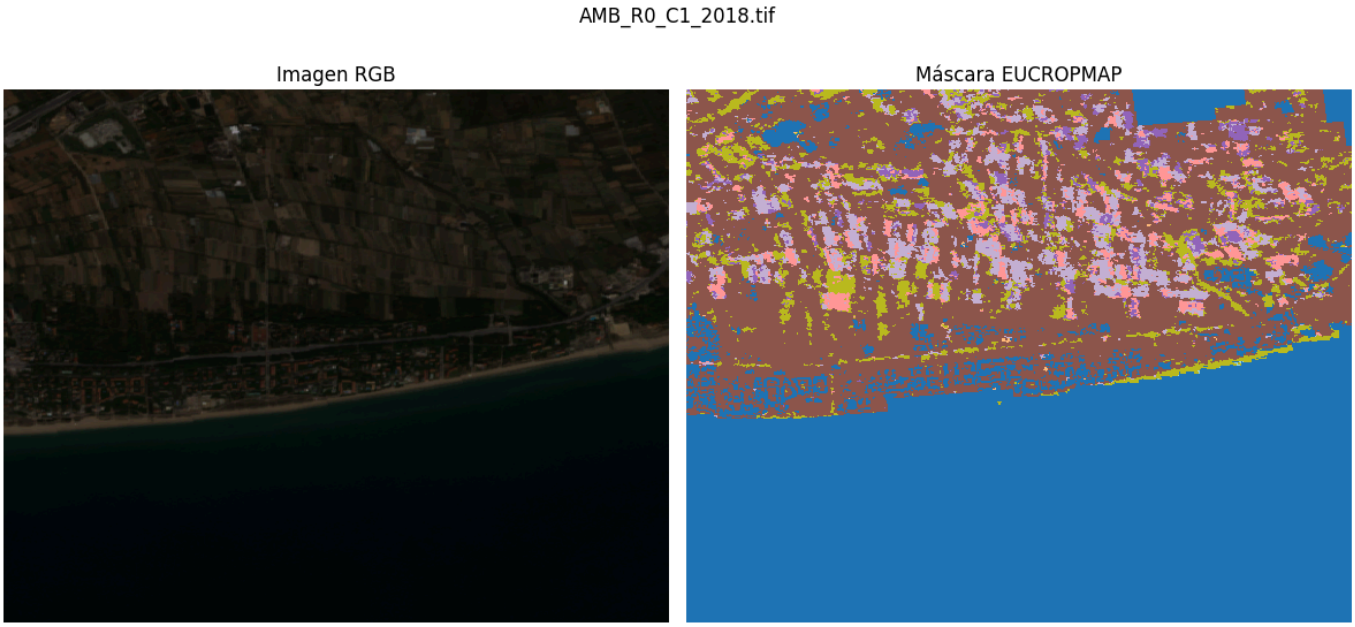
In [5]:
```
import os
import numpy as np
import rasterio
import matplotlib.pyplot as plt
from glob import glob

folder = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB/"
tif_files = sorted(glob(os.path.join(folder, 'AMB_*.tif')))

# Importar librerías necesarias
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import cv2
import matplotlib.pyplot as plt

# Configuración inicial
IMAGE_SIZE = 256  # Tamaño al que redimensionaremos las imágenes y máscaras
BATCH_SIZE = 20  # Tamaño del lote
EPOCHS = 50 # Número de épocas
LEARNING_RATE = 1e-4 # Tasa de aprendizaje
```

2. Función para visualizar RGB + máscara

In [6]:
```
def plot_image_with_cropmask(tif_path):
    with rasterio.open(tif_path) as src:
        img = src.read()  # (bands, rows, cols)
        profile = src.profile

    # Asignar bandas RGB
    rgb = np.stack([img[0], img[1], img[2]], axis=-1)  # B4, B3, B2
    rgb = np.clip(rgb, 0, 10000) / 10000  # Normalizar

    crop = img[3]  # Banda EUCROPMAP

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
    ax1.imshow(rgb)
    ax1.set_title('Imagen RGB')
    ax1.axis('off')

    ax2.imshow(crop, cmap='tab20', interpolation='none')
    ax2.set_title('Máscara EUCROPMAP')
    ax2.axis('off')

    plt.suptitle(os.path.basename(tif_path))
    plt.tight_layout()
    plt.show()
```

3. Visualizar todas

In [ ]:
```
if tif_files:
    plot_image_with_cropmask(tif_files[0])
```



4. Análisis estadístico por celda (EUCROPMAP)

In [ ]:
```
import pandas as pd

# Lista para guardar los resultados
estadisticas = []
imagenes = []

# Procesar cada imagen
for tif in tif_files:
    with rasterio.open(tif) as src:
        img = src.read()
        imagenes.append((os.path.basename(tif), img))

    crop = img[3]
    clases, counts = np.unique(crop, return_counts=True) # Obtener las clases
    total = counts.sum()

    for clase, count in zip(clases, counts):
        estadisticas.append({
            'archivo': os.path.basename(tif),
            'clase': int(clase), # Clase de la máscara EUCROPMAP
            'pixeles': int(count), # Número de píxeles
            'porcentaje': round(100 * count / total, 2) # Porcentaje de píxeles
        })

# Crear DataFrame
df_stats = pd.DataFrame(estadisticas)
df_stats.head()
```

Out[ ]:

	archivo	clase	pixeles	porcentaje
0	AMB_R0_C1_2018.tif	0	122386	49.07
1	AMB_R0_C1_2018.tif	100	75	0.03
2	AMB_R0_C1_2018.tif	211	874	0.35
3	AMB_R0_C1_2018.tif	212	929	0.37
4	AMB_R0_C1_2018.tif	213	2091	0.84

5. Tabla resumen de clases por imagen (en %)

In [ ]:
```
df_resumen = df_stats.pivot_table(index='archivo', columns='clase', values='porcentaje', fill
df_resumen = df_resumen.round(2)

# Mostrar el DataFrame resumen
df_resumen
```

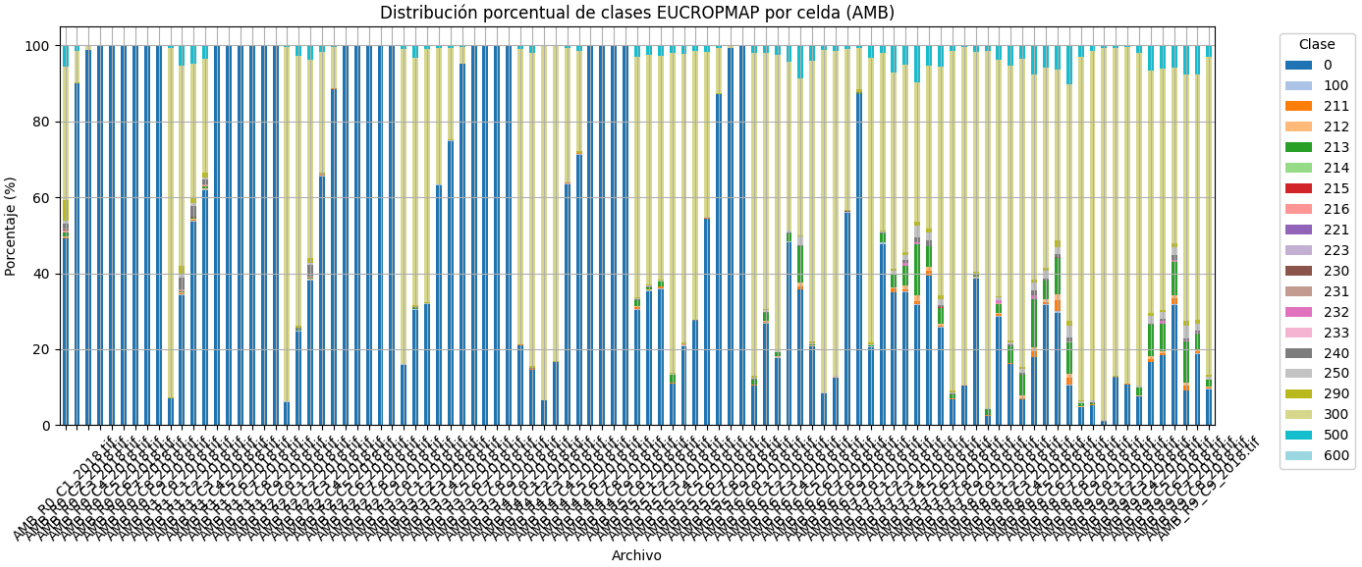
Out[ ]:

clase	0	100	211	212	213	214	215	216	221	223	230	231	232	233
archivo														
AMB_R0_C1_2018.tif	49.07	0.03	0.35	0.37	0.84	0.0	0.00	0.48	0.01	0.0	0.36	0.32	0.00	0.0
AMB_R0_C2_2018.tif	90.05	0.00	0.01	0.01	0.01	0.0	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0
AMB_R0_C3_2018.tif	98.82	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0
AMB_R0_C4_2018.tif	100.00	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0
AMB_R0_C5_2018.tif	100.00	0.00	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0	0.00	0.00	0.00	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
AMB_R9_C5_2018.tif	18.33	0.02	0.61	0.53	7.15	0.0	0.07	0.01	0.00	0.0	0.00	0.03	0.65	0.0
AMB_R9_C6_2018.tif	31.73	0.28	1.30	0.88	8.90	0.0	0.00	0.01	0.00	0.0	0.00	0.03	0.38	0.0
AMB_R9_C7_2018.tif	9.15	0.06	1.17	0.75	10.99	0.0	0.01	0.03	0.00	0.0	0.00	0.00	0.15	0.0
AMB_R9_C8_2018.tif	18.79	0.05	0.70	0.51	4.06	0.0	0.00	0.01	0.00	0.0	0.00	0.00	0.20	0.0
AMB_R9_C9_2018.tif	9.51	0.03	0.31	0.26	1.76	0.0	0.00	0.02	0.00	0.0	0.00	0.01	0.02	0.0

99 rows × 20 columns

6. Gráfico de distribución por clase

In [9]:
```
df_resumen.plot(kind='bar', stacked=True, figsize=(14, 6), colormap='tab20')
plt.title("Distribución porcentual de clases EUCROPMAP por celda (AMB)")
plt.ylabel("Porcentaje (%)")
plt.xlabel("Archivo")
plt.xticks(rotation=45)
plt.legend(title='Clase', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```



7. Análisis temporal por celda

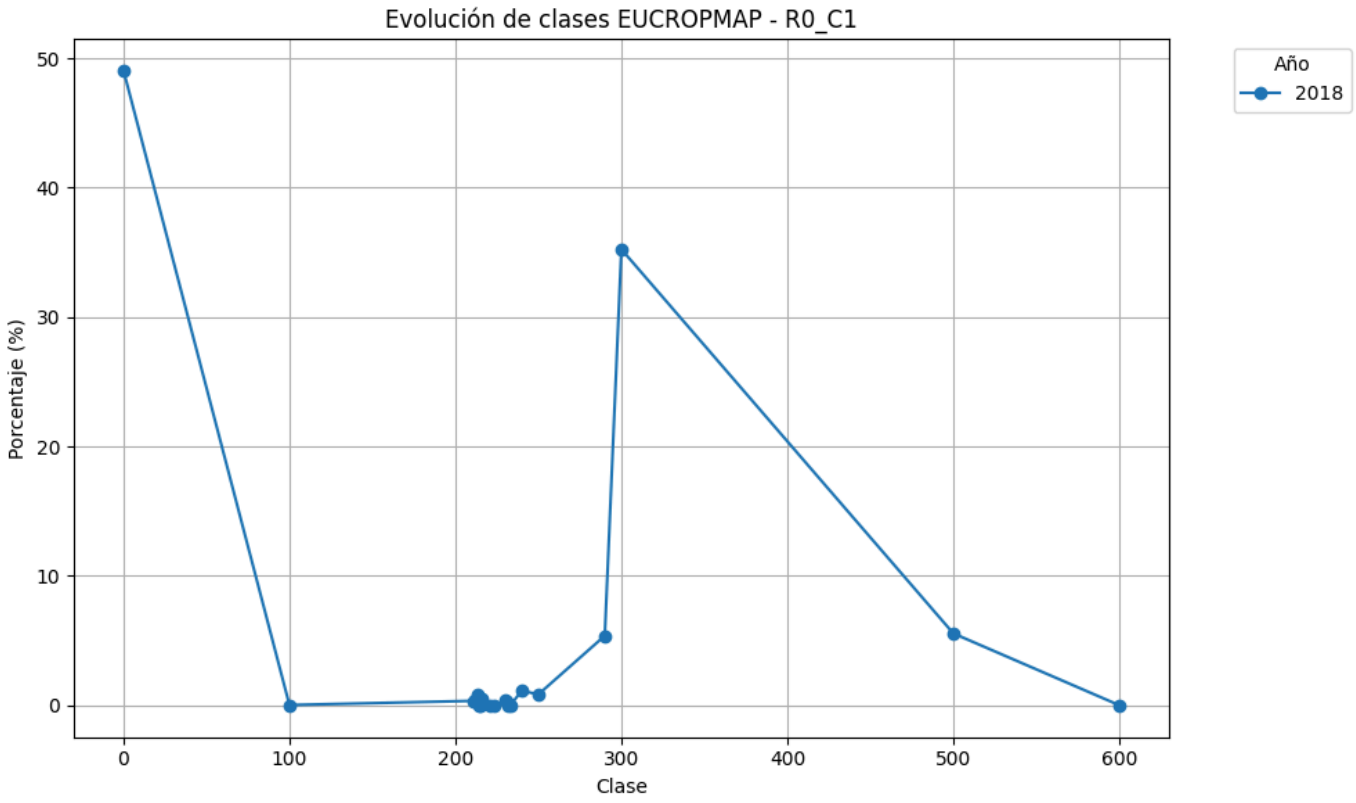
In [10]:
```
# Extraer celda y año del nombre del archivo
df_stats['celda'] = df_stats['archivo'].str.extract(r'(R\d+_C\d+)')
df_stats['anio'] = df_stats['archivo'].str.extract(r'_(\d{4})')

# Pivot para ver cómo cambia cada clase en el tiempo por celda
pivot_tiempo = df_stats.pivot_table(index=['celda', 'anio'], columns='clase', values='porcent
pivot_tiempo.head()
```

Out[10]:

clase		0	100	211	212	213	214	215	216	221	223	230	231	232	233	240	25
celda	**anio**																
R0_C1	**2018**	49.07	0.03	0.35	0.37	0.84	0.0	0.0	0.48	0.01	0.0	0.36	0.32	0.0	0.0	1.13	0.8
R0_C2	**2018**	90.05	0.00	0.01	0.01	0.01	0.0	0.0	0.00	0.00	0.0	0.00	0.00	0.0	0.0	0.00	0.0
R0_C3	**2018**	98.82	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.0	0.00	0.00	0.0	0.0	0.00	0.0
R0_C4	**2018**	100.00	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.0	0.00	0.00	0.0	0.0	0.00	0.0
R0_C5	**2018**	100.00	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.0	0.00	0.00	0.0	0.0	0.00	0.0

8. Visualizar evolución de una celda específica

In [11]:
```
# Elegir una celda para analizar (por ejemplo 'R0_C0')
celda_objetivo = 'R0_C1'

# Comprobar si la celda está disponible
if celda_objetivo in pivot_tiempo.index.get_level_values('celda'):
    pivot_tiempo.loc[celda_objetivo].T.plot(figsize=(10, 6), marker='o')
    plt.title(f"Evolución de clases EUCROPMAP - {celda_objetivo}")
    plt.ylabel("Porcentaje (%)")
    plt.xlabel("Clase")
    plt.grid(True)
    plt.legend(title="Año", bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.show()
else:
    print(f"La celda {celda_objetivo} no está disponible en los datos.")
```



MODELOS

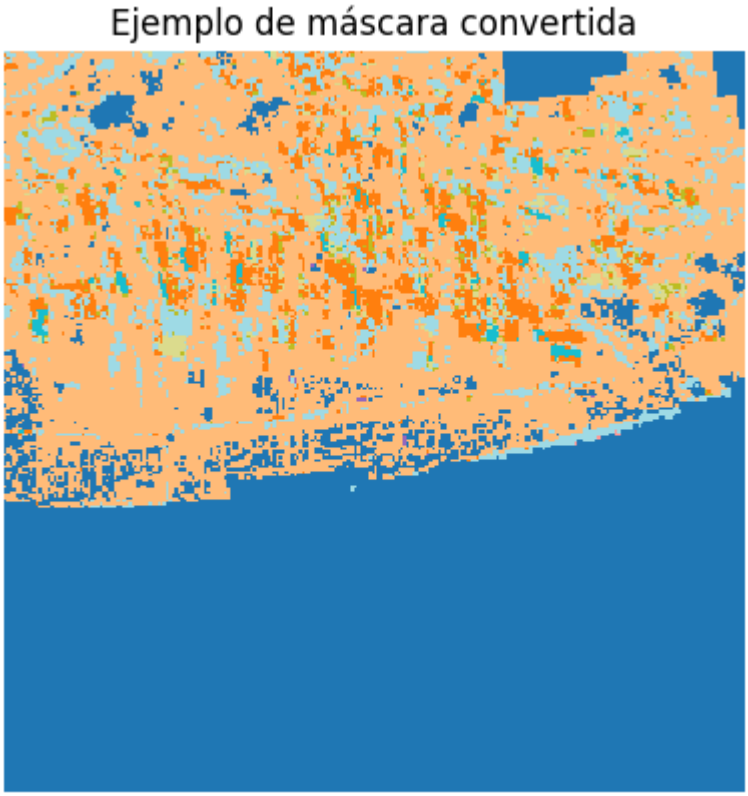
Paso 1: Preparación del dataset

In [12]:
```python
import numpy as np
import rasterio
import os
from glob import glob
from skimage.transform import resize

# Carpeta
# Configuración de directorios
IMAGENES_DIR = r"C:/Users/crome/Desktop/VIU/Code/Datos/AMB/images"
MASCARAS_DIR = r"C:/Users/crome/Desktop/VIU/Code/Datos/AMB/masks"

tif_files = sorted(glob("C:/Users/crome/Desktop/VIU/Code/Datos/AMB/*.tif"))
out_dir = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB/npy_files/"

os.makedirs(IMAGENES_DIR, exist_ok=True)
os.makedirs(MASCARAS_DIR, exist_ok=True)


target_rgb_shape = (256, 256, 3)
target_mask_shape = (256, 256)

errores = []

for tif_path in tif_files:
    name = os.path.basename(tif_path).replace('.tif', '')

    try:
        with rasterio.open(tif_path) as src:
            img = src.read()

        rgb = np.stack([img[0], img[1], img[2]], axis=-1).astype(np.float32) / 10000.0
        mask = img[3].astype(np.uint8)

        # Redimensionar
```

```
        rgb = resize(rgb, target_rgb_shape, preserve_range=True, anti_aliasing=True)
        mask = resize(mask, target_mask_shape, order=0, preserve_range=True).astype(np.uint8)

        np.save(os.path.join(IMAGENES_DIR, f'{name}_rgb.npy'), rgb)
        np.save(os.path.join(MASCARAS_DIR, f'{name}_mask.npy'), mask)

    except Exception as e:
        print(f"Error procesando {name}: {e}")
        errores.append(name)

print("Conversión completada.")
if errores:
    print("Errores en archivos:", errores)
```

```
Conversión completada.
```

In [16]:
```
# Imprimir ejemplo de un archivo convertido
example_file = os.path.join(IMAGENES_DIR, 'AMB_R0_C1_2018_rgb.npy')
if os.path.exists(example_file):
    example_rgb = np.load(example_file)
    plt.imshow(example_rgb)
    plt.title('Ejemplo de imagen RGB convertida')
    plt.axis('off')
    plt.show()
# Imprimir ejemplo de una máscara convertida
example_mask_file = os.path.join(MASCARAS_DIR, 'AMB_R0_C1_2018_mask.npy')
if os.path.exists(example_mask_file):
    example_mask = np.load(example_mask_file)
    plt.imshow(example_mask, cmap='tab20', interpolation='none')
    plt.title('Ejemplo de máscara convertida')
    plt.axis('off')
    plt.show()
```



Paso 2: Definir arquitectura U-Net en Keras

In [21]:

```
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Conv2DTranspose, concatenate
from tensorflow.keras.optimizers import Adam

def unet_model(input_size=(256, 256, 3), num_classes=10):
    inputs = Input(input_size)

    # Encoder
    c1 = Conv2D(32, (3,3), activation='relu', padding='same')(inputs)
    c1 = Conv2D(32, (3,3), activation='relu', padding='same')(c1)
    p1 = MaxPooling2D((2,2))(c1)

    c2 = Conv2D(64, (3,3), activation='relu', padding='same')(p1)
    c2 = Conv2D(64, (3,3), activation='relu', padding='same')(c2)
    p2 = MaxPooling2D((2,2))(c2)

    c3 = Conv2D(128, (3,3), activation='relu', padding='same')(p2)
    c3 = Conv2D(128, (3,3), activation='relu', padding='same')(c3)
    p3 = MaxPooling2D((2,2))(c3)

    # Bottleneck
    c4 = Conv2D(256, (3,3), activation='relu', padding='same')(p3)
    c4 = Conv2D(256, (3,3), activation='relu', padding='same')(c4)

    # Decoder
    u5 = Conv2DTranspose(128, (2,2), strides=(2,2), padding='same')(c4)
    u5 = concatenate([u5, c3])
    c5 = Conv2D(128, (3,3), activation='relu', padding='same')(u5)
    c5 = Conv2D(128, (3,3), activation='relu', padding='same')(c5)

    u6 = Conv2DTranspose(64, (2,2), strides=(2,2), padding='same')(c5)
    u6 = concatenate([u6, c2])
    c6 = Conv2D(64, (3,3), activation='relu', padding='same')(u6)
    c6 = Conv2D(64, (3,3), activation='relu', padding='same')(c6)

    u7 = Conv2DTranspose(32, (2,2), strides=(2,2), padding='same')(c6)
    u7 = concatenate([u7, c1])
    c7 = Conv2D(32, (3,3), activation='relu', padding='same')(u7)
```

```
    c7 = Conv2D(32, (3,3), activation='relu', padding='same')(c7)

    outputs = Conv2D(num_classes, (1,1), activation='softmax')(c7)

    model = Model(inputs=[inputs], outputs=[outputs])
    model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy', metric

    return model
```

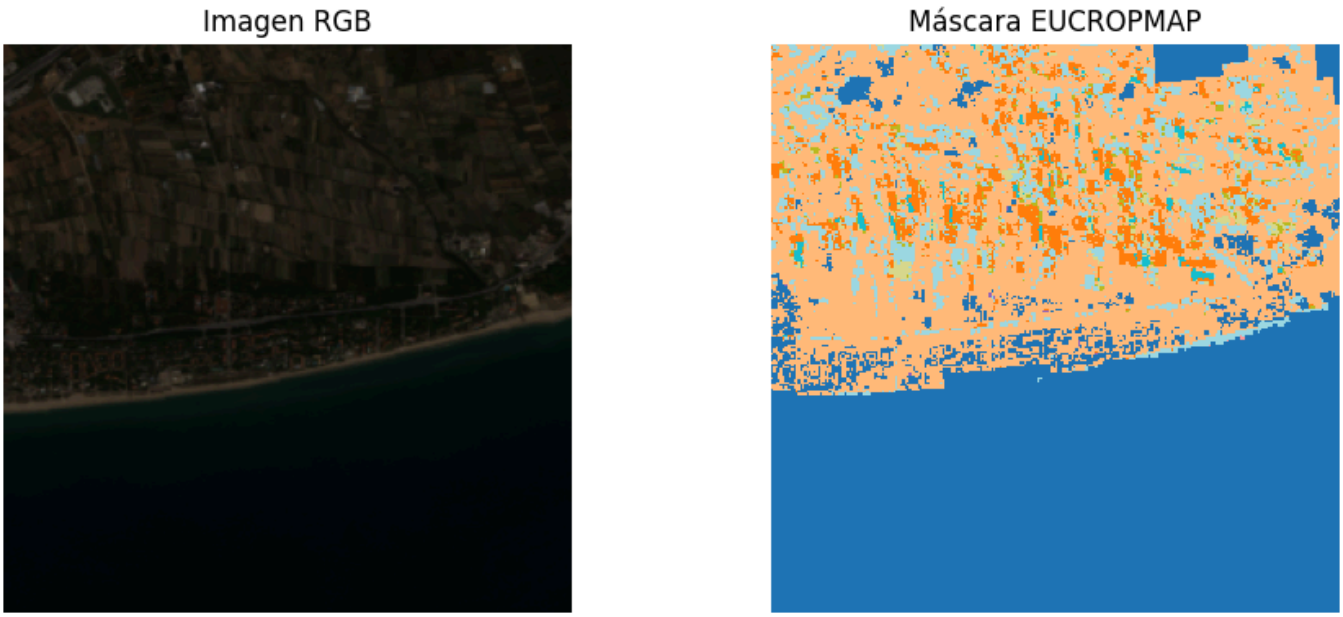
Paso 3: Preparación de los datos (X e y)

In [24]:

```
import numpy as np
import os
from glob import glob
from skimage.transform import resize

# Rutas
folder = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB/unet/"
os.makedirs(folder, exist_ok=True)

rgb_files = sorted(glob(os.path.join(IMAGENES_DIR, '*_rgb.npy')))
mask_files = sorted(glob(os.path.join(MASCARAS_DIR, '*_mask.npy')))

X = []
y = []

# Tamaños estándar
target_rgb_shape = (256, 256, 3)
target_mask_shape = (256, 256)

for rgb_file, mask_file in zip(rgb_files, mask_files):
    # Cargar imagen RGB
    rgb = np.load(rgb_file).astype(np.float32)

    # Normalización automática
    max_val = rgb.max()
    if max_val > 1000:
        rgb /= 10000.0
    elif max_val > 1.1:
        rgb /= 255.0
    # Si ya está entre 0-1, no se hace nada

    # Cargar máscara
    mask = np.load(mask_file).astype(np.uint8)

    # Redimensionar si es necesario
    if rgb.shape != target_rgb_shape:
        rgb = resize(rgb, target_rgb_shape, preserve_range=True, anti_aliasing=True)
    if mask.shape != target_mask_shape:
        mask = resize(mask, target_mask_shape, preserve_range=True, order=0).astype(np.uint8)

    X.append(rgb)
    y.append(mask)

# Convertir a arrays finales
X = np.stack(X)
y = np.stack(y)

# Guardar
np.save(os.path.join(folder, 'X_AMB_2018.npy'), X)
np.save(os.path.join(folder, 'y_AMB_2018.npy'), y)

print("Datos combinados y guardados:")
print("X shape:", X.shape)
print("y shape:", y.shape)
```

```
Datos combinados y guardados:
X shape: (99, 256, 256, 3)
y shape: (99, 256, 256)
```

In [25]:
```python
import matplotlib.pyplot as plt
import numpy as np
import os

# Ruta donde se han guardado
folder = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB/unet/"

# Cargar arrays combinados
X = np.load(os.path.join(folder, 'X_AMB_2018.npy'))
y = np.load(os.path.join(folder, 'y_AMB_2018.npy'))

# Ver ejemplo (por ejemplo el primero)
i = 0  # índice de la imagen que quieres ver

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.imshow(X[i])
plt.title('Imagen RGB')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(y[i], cmap='tab20', interpolation='none')
plt.title('Máscara EUCROPMAP')
plt.axis('off')

plt.tight_layout()
plt.show()
```



Paso 4: Cargar los datos y preparar el entorno

In [2]:
```python
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split

# Cargar datos
folder = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB/unet/"
X = np.load(folder + 'X_AMB_2018.npy')
y = np.load(folder + 'y_AMB_2018.npy')

# Asegurar que esté en formato int
y = y.astype(np.int32)
```

```
# Crear un mapeo de etiquetas únicas a índices continuos
clases_unicas = np.unique(y)
clase_a_indice = {clase: idx for idx, clase in enumerate(clases_unicas)}
indice_a_clase = {idx: clase for clase, idx in clase_a_indice.items()}

# Aplicar mapeo
y_reindex = np.vectorize(clase_a_indice.get)(y)

# Convertir a one-hot
n_classes = len(clases_unicas)
y_cat = tf.keras.utils.to_categorical(y_reindex, num_classes=n_classes)

# División
X_train, X_val, y_train, y_val = train_test_split(X, y_cat, test_size=0.2, random_state=42)

print("Clases mapeadas:", clase_a_indice)
print("X_train:", X_train.shape, "y_train:", y_train.shape)
```

```
Clases mapeadas: {0: 0, 34: 1, 44: 2, 88: 3, 100: 4, 211: 5, 212: 6, 213: 7, 215: 8, 216: 9, 2
21: 10, 223: 11, 230: 12, 231: 13, 232: 14, 240: 15, 244: 16, 250: 17}
X_train: (79, 256, 256, 3) y_train: (79, 256, 256, 18)
```

Paso 5: Entrenamiento del modelo

In [ ]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

model = unet_model(input_size=(256, 256, 3), num_classes=n_classes)

# Callbacks
checkpoint = ModelCheckpoint("mejor_modelo.h5", monitor="val_loss", save_best_only=True)
early_stop = EarlyStopping(monitor="val_loss", patience=10, restore_best_weights=True)

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    batch_size=8,
    epochs=100,
    callbacks=[checkpoint, early_stop]
)
```

```
Epoch 1/100
10/10 [==============================] - 29s 3s/step - loss: 2.8846 - accuracy: 0.3880 - val_l
oss: 2.8760 - val_accuracy: 0.5786
Epoch 2/100
10/10 [==============================] - 27s 3s/step - loss: 2.8710 - accuracy: 0.5461 - val_l
oss: 2.8556 - val_accuracy: 0.4463
Epoch 3/100
10/10 [==============================] - 28s 3s/step - loss: 2.8394 - accuracy: 0.4901 - val_l
oss: 2.7793 - val_accuracy: 0.4675
Epoch 4/100
10/10 [==============================] - 26s 3s/step - loss: 2.5579 - accuracy: 0.5000 - val_l
oss: 1.7456 - val_accuracy: 0.4666
Epoch 5/100
10/10 [==============================] - 25s 3s/step - loss: 1.4529 - accuracy: 0.4702 - val_l
oss: 1.5070 - val_accuracy: 0.4585
Epoch 6/100
10/10 [==============================] - 25s 3s/step - loss: 1.1854 - accuracy: 0.4596 - val_l
oss: 1.2860 - val_accuracy: 0.4681
Epoch 7/100
10/10 [==============================] - 25s 3s/step - loss: 1.1460 - accuracy: 0.5008 - val_l
oss: 1.2841 - val_accuracy: 0.4677
Epoch 8/100
10/10 [==============================] - 25s 3s/step - loss: 1.0998 - accuracy: 0.4851 - val_l
oss: 1.2058 - val_accuracy: 0.4630
Epoch 9/100
10/10 [==============================] - 26s 3s/step - loss: 1.0440 - accuracy: 0.4936 - val_l
oss: 1.1891 - val_accuracy: 0.4724
Epoch 10/100
10/10 [==============================] - 25s 3s/step - loss: 1.0173 - accuracy: 0.5170 - val_l
oss: 1.1425 - val_accuracy: 0.4694
Epoch 11/100
10/10 [==============================] - 27s 3s/step - loss: 0.9810 - accuracy: 0.5155 - val_l
oss: 1.0842 - val_accuracy: 0.4686
Epoch 12/100
10/10 [==============================] - 25s 3s/step - loss: 0.9554 - accuracy: 0.5141 - val_l
oss: 1.0549 - val_accuracy: 0.4691
Epoch 13/100
10/10 [==============================] - 36s 4s/step - loss: 0.9298 - accuracy: 0.5166 - val_l
oss: 1.0381 - val_accuracy: 0.4697
Epoch 14/100
10/10 [==============================] - 41s 4s/step - loss: 0.9019 - accuracy: 0.5160 - val_l
oss: 1.0091 - val_accuracy: 0.4691
Epoch 15/100
10/10 [==============================] - 41s 4s/step - loss: 0.8633 - accuracy: 0.5168 - val_l
oss: 0.9858 - val_accuracy: 0.4700
Epoch 16/100
10/10 [==============================] - 43s 4s/step - loss: 0.8189 - accuracy: 0.5170 - val_l
oss: 0.9554 - val_accuracy: 0.4700
Epoch 17/100
10/10 [==============================] - 41s 4s/step - loss: 0.7755 - accuracy: 0.5171 - val_l
oss: 0.9302 - val_accuracy: 0.4697
Epoch 18/100
10/10 [==============================] - 45s 5s/step - loss: 0.7423 - accuracy: 0.5167 - val_l
oss: 0.9069 - val_accuracy: 0.4705
Epoch 19/100
10/10 [==============================] - 42s 4s/step - loss: 0.7251 - accuracy: 0.5173 - val_l
oss: 0.8940 - val_accuracy: 0.4703
Epoch 20/100
10/10 [==============================] - 41s 4s/step - loss: 0.7121 - accuracy: 0.5161 - val_l
oss: 0.8944 - val_accuracy: 0.4701
Epoch 21/100
10/10 [==============================] - 41s 4s/step - loss: 0.7072 - accuracy: 0.5179 - val_l
oss: 0.8806 - val_accuracy: 0.4585
Epoch 22/100
10/10 [==============================] - 41s 4s/step - loss: 0.7323 - accuracy: 0.5203 - val_l
oss: 0.9199 - val_accuracy: 0.4777
```

```
Epoch 23/100
10/10 [==============================] - 41s 4s/step - loss: 0.7340 - accuracy: 0.5200 - val_l
oss: 0.9098 - val_accuracy: 0.4807
Epoch 24/100
10/10 [==============================] - 40s 4s/step - loss: 0.7056 - accuracy: 0.5219 - val_l
oss: 0.8765 - val_accuracy: 0.4770
Epoch 25/100
10/10 [==============================] - 42s 4s/step - loss: 0.6897 - accuracy: 0.5429 - val_l
oss: 0.8681 - val_accuracy: 0.4989
Epoch 26/100
10/10 [==============================] - 40s 4s/step - loss: 0.6799 - accuracy: 0.5663 - val_l
oss: 0.8692 - val_accuracy: 0.5619
Epoch 27/100
10/10 [==============================] - 41s 4s/step - loss: 0.6724 - accuracy: 0.6343 - val_l
oss: 0.8546 - val_accuracy: 0.5955
Epoch 28/100
10/10 [==============================] - 41s 4s/step - loss: 0.6812 - accuracy: 0.6585 - val_l
oss: 0.9453 - val_accuracy: 0.4702
Epoch 29/100
10/10 [==============================] - 41s 4s/step - loss: 0.7909 - accuracy: 0.5175 - val_l
oss: 0.9447 - val_accuracy: 0.4702
Epoch 30/100
10/10 [==============================] - 41s 4s/step - loss: 0.7515 - accuracy: 0.5570 - val_l
oss: 0.8981 - val_accuracy: 0.5232
Epoch 31/100
10/10 [==============================] - 43s 4s/step - loss: 0.7038 - accuracy: 0.6209 - val_l
oss: 0.8575 - val_accuracy: 0.5999
Epoch 32/100
10/10 [==============================] - 41s 4s/step - loss: 0.6661 - accuracy: 0.7222 - val_l
oss: 0.8509 - val_accuracy: 0.6057
Epoch 33/100
10/10 [==============================] - 41s 4s/step - loss: 0.6595 - accuracy: 0.7024 - val_l
oss: 0.8659 - val_accuracy: 0.6219
Epoch 34/100
10/10 [==============================] - 42s 4s/step - loss: 0.6575 - accuracy: 0.7515 - val_l
oss: 0.8493 - val_accuracy: 0.6043
Epoch 35/100
10/10 [==============================] - 41s 4s/step - loss: 0.6625 - accuracy: 0.6860 - val_l
oss: 0.8407 - val_accuracy: 0.6347
Epoch 36/100
10/10 [==============================] - 41s 4s/step - loss: 0.6453 - accuracy: 0.7287 - val_l
oss: 0.8232 - val_accuracy: 0.6870
Epoch 37/100
10/10 [==============================] - 41s 4s/step - loss: 0.6157 - accuracy: 0.7835 - val_l
oss: 0.8065 - val_accuracy: 0.6912
Epoch 38/100
10/10 [==============================] - 41s 4s/step - loss: 0.5961 - accuracy: 0.7854 - val_l
oss: 0.7941 - val_accuracy: 0.6906
Epoch 39/100
10/10 [==============================] - 41s 4s/step - loss: 0.5830 - accuracy: 0.7861 - val_l
oss: 0.7836 - val_accuracy: 0.6926
Epoch 40/100
10/10 [==============================] - 42s 4s/step - loss: 0.5729 - accuracy: 0.7884 - val_l
oss: 0.7794 - val_accuracy: 0.6937
Epoch 41/100
10/10 [==============================] - 41s 4s/step - loss: 0.5608 - accuracy: 0.7890 - val_l
oss: 0.7645 - val_accuracy: 0.6947
Epoch 42/100
10/10 [==============================] - 41s 4s/step - loss: 0.5530 - accuracy: 0.7896 - val_l
oss: 0.7616 - val_accuracy: 0.6936
Epoch 43/100
10/10 [==============================] - 41s 4s/step - loss: 0.5554 - accuracy: 0.7882 - val_l
oss: 0.7522 - val_accuracy: 0.6951
Epoch 44/100
10/10 [==============================] - 41s 4s/step - loss: 0.5468 - accuracy: 0.7890 - val_l
oss: 0.7520 - val_accuracy: 0.6946
```

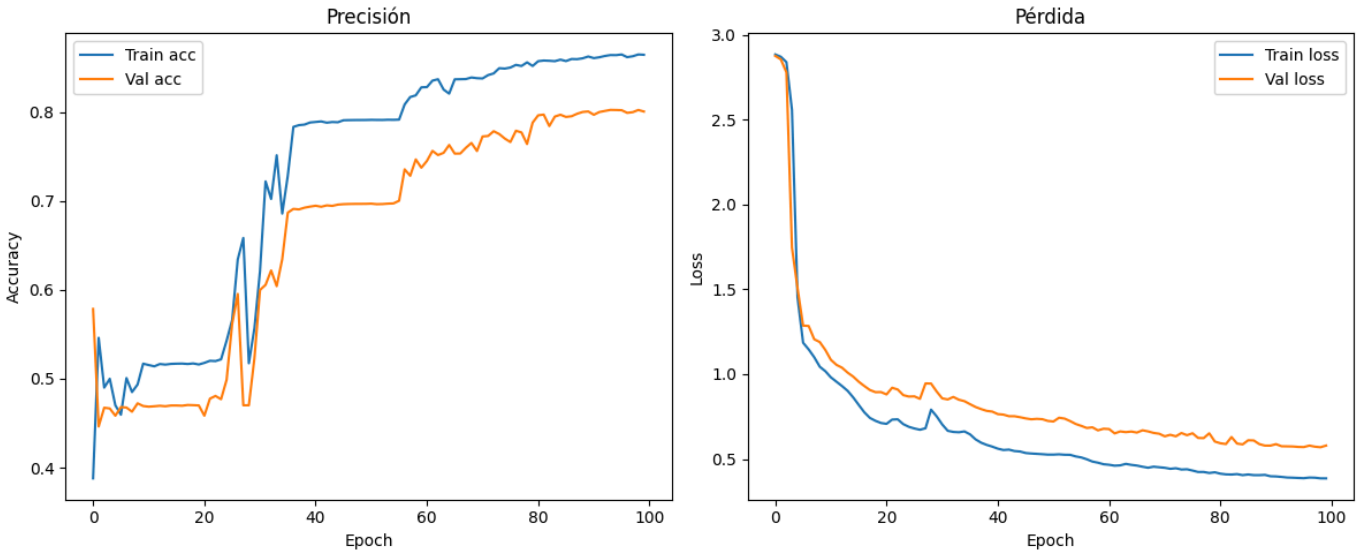
```
Epoch 45/100
10/10 [==============================] - 41s 4s/step - loss: 0.5442 - accuracy: 0.7887 - val_l
oss: 0.7462 - val_accuracy: 0.6960
Epoch 46/100
10/10 [==============================] - 41s 4s/step - loss: 0.5352 - accuracy: 0.7909 - val_l
oss: 0.7393 - val_accuracy: 0.6965
Epoch 47/100
10/10 [==============================] - 42s 4s/step - loss: 0.5323 - accuracy: 0.7911 - val_l
oss: 0.7341 - val_accuracy: 0.6967
Epoch 48/100
10/10 [==============================] - 41s 4s/step - loss: 0.5302 - accuracy: 0.7912 - val_l
oss: 0.7366 - val_accuracy: 0.6968
Epoch 49/100
10/10 [==============================] - 41s 4s/step - loss: 0.5283 - accuracy: 0.7912 - val_l
oss: 0.7347 - val_accuracy: 0.6969
Epoch 50/100
10/10 [==============================] - 41s 4s/step - loss: 0.5259 - accuracy: 0.7912 - val_l
oss: 0.7244 - val_accuracy: 0.6969
Epoch 51/100
10/10 [==============================] - 41s 4s/step - loss: 0.5259 - accuracy: 0.7914 - val_l
oss: 0.7212 - val_accuracy: 0.6971
Epoch 52/100
10/10 [==============================] - 40s 4s/step - loss: 0.5276 - accuracy: 0.7913 - val_l
oss: 0.7431 - val_accuracy: 0.6965
Epoch 53/100
10/10 [==============================] - 40s 4s/step - loss: 0.5250 - accuracy: 0.7913 - val_l
oss: 0.7382 - val_accuracy: 0.6967
Epoch 54/100
10/10 [==============================] - 41s 4s/step - loss: 0.5246 - accuracy: 0.7915 - val_l
oss: 0.7234 - val_accuracy: 0.6971
Epoch 55/100
10/10 [==============================] - 42s 4s/step - loss: 0.5152 - accuracy: 0.7914 - val_l
oss: 0.7064 - val_accuracy: 0.6974
Epoch 56/100
10/10 [==============================] - 41s 4s/step - loss: 0.5092 - accuracy: 0.7917 - val_l
oss: 0.6952 - val_accuracy: 0.7004
Epoch 57/100
10/10 [==============================] - 41s 4s/step - loss: 0.4982 - accuracy: 0.8088 - val_l
oss: 0.6827 - val_accuracy: 0.7357
Epoch 58/100
10/10 [==============================] - 41s 4s/step - loss: 0.4849 - accuracy: 0.8171 - val_l
oss: 0.6867 - val_accuracy: 0.7285
Epoch 59/100
10/10 [==============================] - 40s 4s/step - loss: 0.4781 - accuracy: 0.8190 - val_l
oss: 0.6687 - val_accuracy: 0.7469
Epoch 60/100
10/10 [==============================] - 41s 4s/step - loss: 0.4692 - accuracy: 0.8280 - val_l
oss: 0.6784 - val_accuracy: 0.7375
Epoch 61/100
10/10 [==============================] - 41s 4s/step - loss: 0.4659 - accuracy: 0.8282 - val_l
oss: 0.6764 - val_accuracy: 0.7451
Epoch 62/100
10/10 [==============================] - 42s 4s/step - loss: 0.4606 - accuracy: 0.8355 - val_l
oss: 0.6509 - val_accuracy: 0.7565
Epoch 63/100
10/10 [==============================] - 41s 4s/step - loss: 0.4625 - accuracy: 0.8371 - val_l
oss: 0.6624 - val_accuracy: 0.7518
Epoch 64/100
10/10 [==============================] - 42s 4s/step - loss: 0.4712 - accuracy: 0.8255 - val_l
oss: 0.6584 - val_accuracy: 0.7543
Epoch 65/100
10/10 [==============================] - 40s 4s/step - loss: 0.4653 - accuracy: 0.8209 - val_l
oss: 0.6617 - val_accuracy: 0.7631
Epoch 66/100
10/10 [==============================] - 41s 4s/step - loss: 0.4613 - accuracy: 0.8370 - val_l
oss: 0.6558 - val_accuracy: 0.7534
```

```
Epoch 67/100
10/10 [==============================] - 40s 4s/step - loss: 0.4544 - accuracy: 0.8372 - val_l
oss: 0.6691 - val_accuracy: 0.7535
Epoch 68/100
10/10 [==============================] - 40s 4s/step - loss: 0.4484 - accuracy: 0.8373 - val_l
oss: 0.6627 - val_accuracy: 0.7600
Epoch 69/100
10/10 [==============================] - 41s 4s/step - loss: 0.4547 - accuracy: 0.8391 - val_l
oss: 0.6537 - val_accuracy: 0.7654
Epoch 70/100
10/10 [==============================] - 42s 4s/step - loss: 0.4513 - accuracy: 0.8383 - val_l
oss: 0.6490 - val_accuracy: 0.7563
Epoch 71/100
10/10 [==============================] - 40s 4s/step - loss: 0.4481 - accuracy: 0.8380 - val_l
oss: 0.6337 - val_accuracy: 0.7727
Epoch 72/100
10/10 [==============================] - 41s 4s/step - loss: 0.4421 - accuracy: 0.8418 - val_l
oss: 0.6427 - val_accuracy: 0.7733
Epoch 73/100
10/10 [==============================] - 41s 4s/step - loss: 0.4454 - accuracy: 0.8436 - val_l
oss: 0.6337 - val_accuracy: 0.7786
Epoch 74/100
10/10 [==============================] - 40s 4s/step - loss: 0.4380 - accuracy: 0.8495 - val_l
oss: 0.6531 - val_accuracy: 0.7754
Epoch 75/100
10/10 [==============================] - 41s 4s/step - loss: 0.4394 - accuracy: 0.8491 - val_l
oss: 0.6403 - val_accuracy: 0.7704
Epoch 76/100
10/10 [==============================] - 41s 4s/step - loss: 0.4318 - accuracy: 0.8502 - val_l
oss: 0.6517 - val_accuracy: 0.7664
Epoch 77/100
10/10 [==============================] - 41s 4s/step - loss: 0.4234 - accuracy: 0.8533 - val_l
oss: 0.6241 - val_accuracy: 0.7791
Epoch 78/100
10/10 [==============================] - 43s 4s/step - loss: 0.4236 - accuracy: 0.8520 - val_l
oss: 0.6230 - val_accuracy: 0.7772
Epoch 79/100
10/10 [==============================] - 41s 4s/step - loss: 0.4175 - accuracy: 0.8561 - val_l
oss: 0.6514 - val_accuracy: 0.7642
Epoch 80/100
10/10 [==============================] - 40s 4s/step - loss: 0.4217 - accuracy: 0.8520 - val_l
oss: 0.6029 - val_accuracy: 0.7882
Epoch 81/100
10/10 [==============================] - 39s 4s/step - loss: 0.4135 - accuracy: 0.8574 - val_l
oss: 0.5927 - val_accuracy: 0.7965
Epoch 82/100
10/10 [==============================] - 40s 4s/step - loss: 0.4094 - accuracy: 0.8581 - val_l
oss: 0.5881 - val_accuracy: 0.7973
Epoch 83/100
10/10 [==============================] - 39s 4s/step - loss: 0.4084 - accuracy: 0.8579 - val_l
oss: 0.6295 - val_accuracy: 0.7844
Epoch 84/100
10/10 [==============================] - 40s 4s/step - loss: 0.4109 - accuracy: 0.8574 - val_l
oss: 0.5910 - val_accuracy: 0.7951
Epoch 85/100
10/10 [==============================] - 40s 4s/step - loss: 0.4049 - accuracy: 0.8591 - val_l
oss: 0.5860 - val_accuracy: 0.7971
Epoch 86/100
10/10 [==============================] - 40s 4s/step - loss: 0.4086 - accuracy: 0.8575 - val_l
oss: 0.6104 - val_accuracy: 0.7947
Epoch 87/100
10/10 [==============================] - 41s 4s/step - loss: 0.4049 - accuracy: 0.8599 - val_l
oss: 0.6092 - val_accuracy: 0.7954
Epoch 88/100
10/10 [==============================] - 41s 4s/step - loss: 0.4048 - accuracy: 0.8597 - val_l
oss: 0.5877 - val_accuracy: 0.7982
```

```
Epoch 89/100
10/10 [==============================] - 41s 4s/step - loss: 0.4064 - accuracy: 0.8607 - val_l
oss: 0.5789 - val_accuracy: 0.8003
Epoch 90/100
10/10 [==============================] - 40s 4s/step - loss: 0.3982 - accuracy: 0.8627 - val_l
oss: 0.5788 - val_accuracy: 0.8009
Epoch 91/100
10/10 [==============================] - 40s 4s/step - loss: 0.3972 - accuracy: 0.8609 - val_l
oss: 0.5875 - val_accuracy: 0.7971
Epoch 92/100
10/10 [==============================] - 41s 4s/step - loss: 0.3943 - accuracy: 0.8618 - val_l
oss: 0.5751 - val_accuracy: 0.8002
Epoch 93/100
10/10 [==============================] - 40s 4s/step - loss: 0.3908 - accuracy: 0.8633 - val_l
oss: 0.5744 - val_accuracy: 0.8014
Epoch 94/100
10/10 [==============================] - 41s 4s/step - loss: 0.3896 - accuracy: 0.8643 - val_l
oss: 0.5737 - val_accuracy: 0.8026
Epoch 95/100
10/10 [==============================] - 40s 4s/step - loss: 0.3882 - accuracy: 0.8642 - val_l
oss: 0.5712 - val_accuracy: 0.8025
Epoch 96/100
10/10 [==============================] - 41s 4s/step - loss: 0.3869 - accuracy: 0.8649 - val_l
oss: 0.5703 - val_accuracy: 0.8023
Epoch 97/100
10/10 [==============================] - 41s 4s/step - loss: 0.3906 - accuracy: 0.8619 - val_l
oss: 0.5789 - val_accuracy: 0.7993
Epoch 98/100
10/10 [==============================] - 41s 4s/step - loss: 0.3897 - accuracy: 0.8630 - val_l
oss: 0.5724 - val_accuracy: 0.8000
Epoch 99/100
10/10 [==============================] - 41s 4s/step - loss: 0.3860 - accuracy: 0.8650 - val_l
oss: 0.5692 - val_accuracy: 0.8025
Epoch 100/100
10/10 [==============================] - 41s 4s/step - loss: 0.3858 - accuracy: 0.8646 - val_l
oss: 0.5791 - val_accuracy: 0.8008
```

In [ ]:
```python
# Guardar el modelo final
model.save("modelo_final.h5")
```

In [40]:
```python
# Gráfica de accuracy y loss
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train acc')
plt.plot(history.history['val_accuracy'], label='Val acc')
plt.title('Precisión')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train loss')
plt.plot(history.history['val_loss'], label='Val loss')
plt.title('Pérdida')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



Paso 4: Visualización de predicciones

In [3]:
```python
# Cargar el mejor modelo
from tensorflow.keras.models import load_model
model = load_model("mejor_modelo.h5")

# Evaluación del Mejor Modelo
loss, accuracy = model.evaluate(X_val, y_val)
print(f"Pérdida en validación: {loss:.4f}")
print(f"Precisión en validación: {accuracy:.4f}")
```

```
1/1 [==============================] - 3s 3s/step - loss: 0.5692 - accuracy: 0.8025
Pérdida en validación: 0.5692
Precisión en validación: 0.8025
1/1 [==============================] - 3s 3s/step - loss: 0.5692 - accuracy: 0.8025
Pérdida en validación: 0.5692
Precisión en validación: 0.8025
```

In [6]:
```python
import numpy as np
from sklearn.metrics import confusion_matrix

def calcular_metricas_segmentacion(y_true, y_pred, n_clases):
    """
    Calcula métricas de segmentación (IoU, Dice y Accuracy) por clase.

    y_true: (N, H, W) ground truth, valores enteros por píxel
    y_pred: (N, H, W) predicción del modelo, valores enteros por píxel
    n_clases: número total de clases (de 0 a n_clases-1)
    """
    assert y_true.shape == y_pred.shape, "Tamaño inconsistente"

    iou_list = []
    dice_list = []
    accuracy_list = []

    print(f"{'Clase':<8} {'IoU':>10} {'Dice':>10} {'Accuracy':>10}")
    print("-" * 40)

    for clase in range(n_clases):
        gt = (y_true == clase)
        pred = (y_pred == clase)

        intersection = np.logical_and(gt, pred).sum()
        union = np.logical_or(gt, pred).sum()
        total_pred = pred.sum()
        total_gt = gt.sum()

        iou = intersection / union if union != 0 else 0.0
```

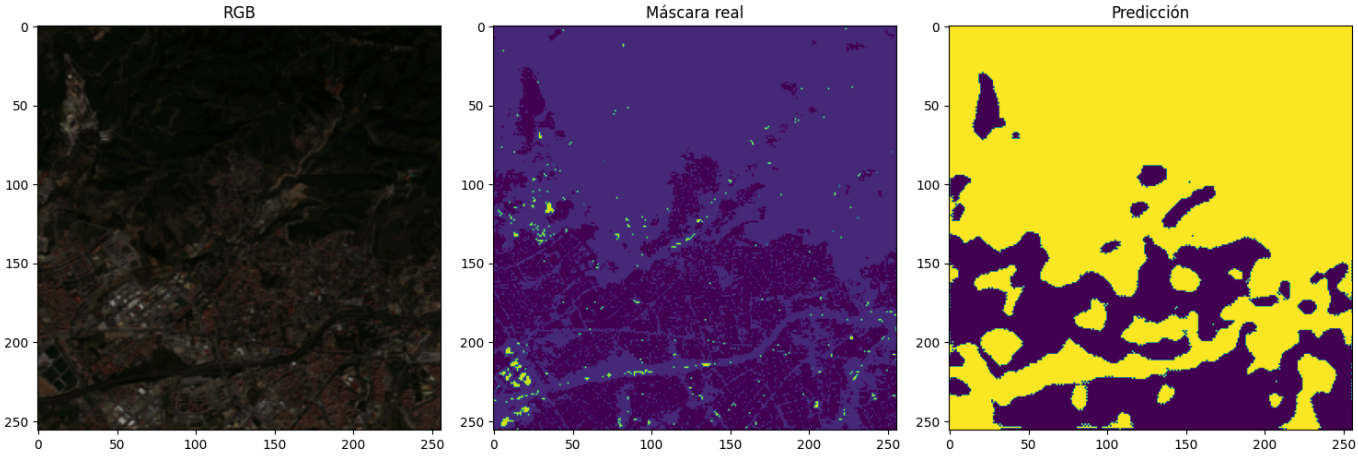
```
        dice = 2 * intersection / (total_pred + total_gt) if (total_pred + total_gt) != 0 els
        accuracy = intersection / total_gt if total_gt != 0 else 0.0

        iou_list.append(iou)
        dice_list.append(dice)
        accuracy_list.append(accuracy)

        print(f"{clase:<8} {iou:10.4f} {dice:10.4f} {accuracy:10.4f}")

    print("-" * 40)
    print(f"{'PROMEDIO':<8} {np.mean(iou_list):10.4f} {np.mean(dice_list):10.4f} {np.mean(acc
    return iou_list, dice_list, accuracy_list


y_val_int = np.argmax(y_val, axis=-1)
y_pred_raw = model.predict(X_val)
y_pred_int = np.argmax(y_pred_raw, axis=-1)
n_clases = len(np.unique(y_val_int))

iou, dice, acc = calcular_metricas_segmentacion(y_val_int, y_pred_int, n_clases)
```

```
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
Clase           IoU       Dice   Accuracy
----------------------------------------
0            0.7587     0.8628     0.8690
1            0.0000     0.0000     0.0000
2            0.6755     0.8063     0.8676
3            0.0000     0.0000     0.0000
4            0.0000     0.0000     0.0000
5            0.0000     0.0000     0.0000
6            0.0000     0.0000     0.0000
7            0.0000     0.0000     0.0000
8            0.0000     0.0000     0.0000
9            0.0000     0.0000     0.0000
10           0.0000     0.0000     0.0000
11           0.0000     0.0000     0.0000
12           0.0000     0.0000     0.0000
13           0.0000     0.0000     0.0000
14           0.0000     0.0000     0.0000
15           0.0000     0.0000     0.0000
16           0.0000     0.0000     0.0000
17           0.0000     0.0000     0.0000
----------------------------------------
PROMEDIO     0.0797     0.0927     0.0965
```

In [25]:

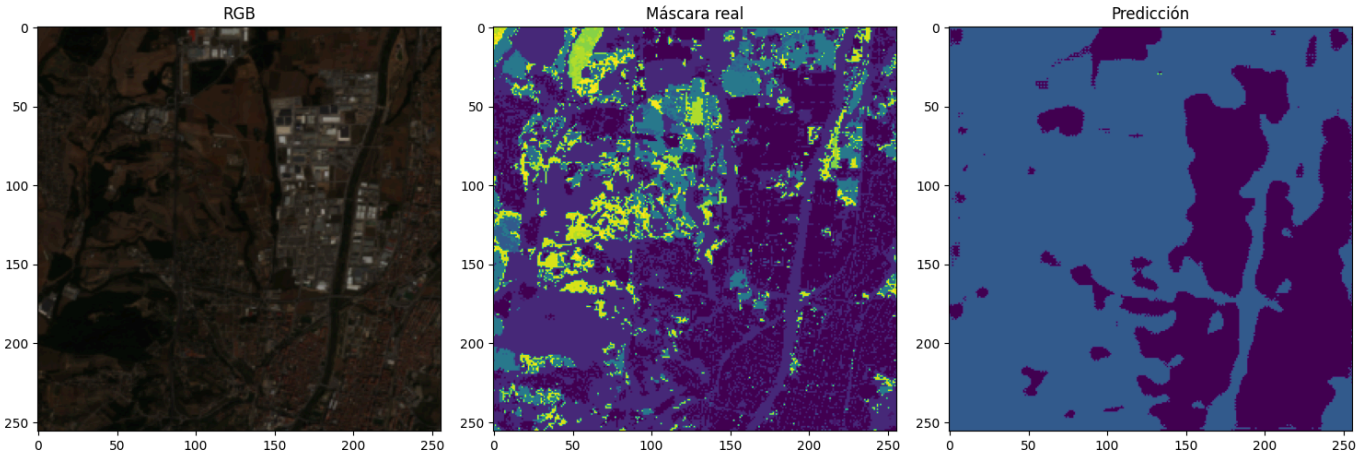
```
# Predicción sobre una imagen del conjunto de validación
i = 10
pred_mask = model.predict(np.expand_dims(X_val[i], axis=0))[0]
pred_mask_argmax = np.argmax(pred_mask, axis=-1)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15,5))
ax1.imshow(X_val[i])
ax1.set_title("RGB")
ax2.imshow(np.argmax(y_val[i], axis=-1))
ax2.set_title("Máscara real")
ax3.imshow(pred_mask_argmax)
ax3.set_title("Predicción")
plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 0s 248ms/step
```



In [26]:
```
# Predicción sobre una imagen del conjunto de validación
i = 2
pred_mask = model.predict(np.expand_dims(X_val[i], axis=0))[0]
pred_mask_argmax = np.argmax(pred_mask, axis=-1)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15,5))
ax1.imshow(X_val[i])
ax1.set_title("RGB")
ax2.imshow(np.argmax(y_val[i], axis=-1))
ax2.set_title("Máscara real")
ax3.imshow(pred_mask_argmax)
ax3.set_title("Predicción")
plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 0s 132ms/step
```



In [ ]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras import backend as K
import pandas as pd
import os
import gc


batch_sizes = [8, 16, 32]
epoch_list = [25, 50, 100]

resultados = []

for batch in batch_sizes:
    for epochs in epoch_list:
        print(f"\n Entrenando con batch_size={batch}, epochs={epochs}")

        # Liberar memoria previa
        K.clear_session()
```

```
        gc.collect()

        # Crear nuevo modelo desde cero
        model = unet_model(input_size=(256, 256, 3), num_classes=n_classes)

        # Callbacks
        nombre_modelo = f"model_bs{batch}_ep{epochs}.h5"
        checkpoint = ModelCheckpoint(nombre_modelo, monitor="val_loss", save_best_only=True,
        early_stop = EarlyStopping(monitor="val_loss", patience=10, restore_best_weights=True

        history = model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            batch_size=batch,
            epochs=epochs,
            callbacks=[checkpoint, early_stop],
            verbose=1
        )

        best_epoch = np.argmin(history.history['val_loss']) + 1
        final_val_loss = np.min(history.history['val_loss'])
        final_val_acc = history.history['val_accuracy'][best_epoch - 1]

        resultados.append({
            'batch_size': batch,
            'epochs': epochs,
            'best_epoch': best_epoch,
            'val_loss': round(final_val_loss, 4),
            'val_accuracy': round(final_val_acc, 4),
            'modelo': nombre_modelo
        })

# Mostrar resultados en tabla ordenada
df_resultados = pd.DataFrame(resultados)
df_resultados = df_resultados.sort_values(by='val_loss')
print("\n Resultados ordenados por menor pérdida en validación:\n")
print(df_resultados)
```

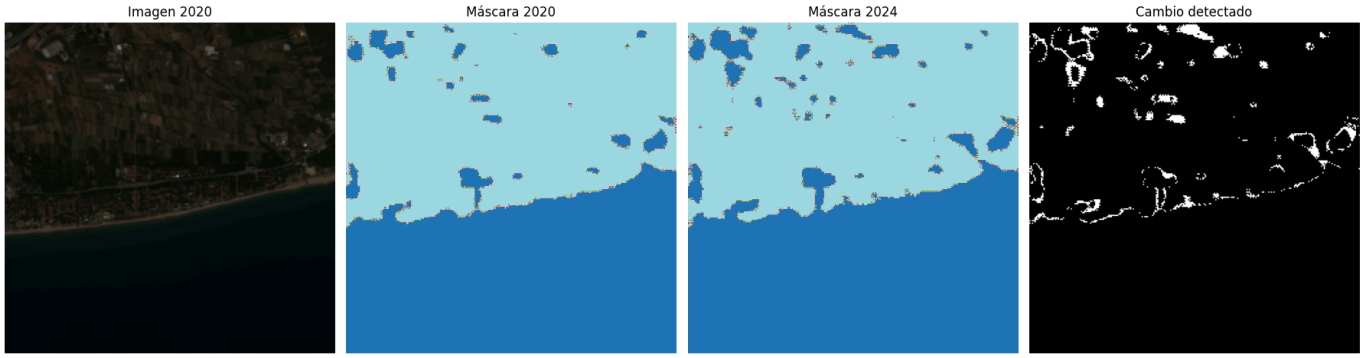
Código para evaluar cambio de suelo

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model

# Cargar modelo entrenado
model = load_model("mejor_modelo.h5")

# Función para predecir la máscara
def predict_mask(img):
    pred = model.predict(np.expand_dims(img, axis=0))[0]
    return np.argmax(pred, axis=-1)

# Cargar dos imágenes de la misma zona (fechas distintas)
img_2020 = np.load("C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/images/AMB_R0_C1_2020_rgb.
img_2023 = np.load("C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/images/AMB_R0_C1_2024_rgb.

# Normalizar si no están ya
if img_2020.max() > 1: img_2020 = img_2020 / 10000.0
if img_2023.max() > 1: img_2023 = img_2023 / 10000.0

# Predecir
pred_2020 = predict_mask(img_2020)
pred_2023 = predict_mask(img_2023)

# Mapa de cambio
```

```
cambio = pred_2020 != pred_2023

# Visualización
fig, axs = plt.subplots(1, 4, figsize=(18, 5))
axs[0].imshow(img_2020)
axs[0].set_title("Imagen 2020")
axs[1].imshow(pred_2020, cmap="tab20")
axs[1].set_title("Máscara 2020")
axs[2].imshow(pred_2023, cmap="tab20")
axs[2].set_title("Máscara 2024")
axs[3].imshow(cambio, cmap='gray')
axs[3].set_title("Cambio detectado")
for ax in axs: ax.axis('off')
plt.tight_layout()
plt.show()

# Estadística
n_cambiados = np.sum(cambio)
total = cambio.size
print(f"Porcentaje de píxeles cambiados: {100 * n_cambiados / total:.2f}%")
```

```
1/1 [==============================] - 0s 280ms/step
1/1 [==============================] - 0s 97ms/step
```




```
Porcentaje de píxeles cambiados: 3.04%
```

In [34]:

```
clase_eucropmap = {
    0:  "Sin información / fondo",
    34: "Cereal – Trigo (soft wheat)",
    44: "Cereal – Cebada (barley)",
    88: "Cereal – Maíz (corn)",
    100: "Cereal – Otros cereales",
    211: "Leguminosas – Guisante (pea)",
    212: "Leguminosas – Alfalfa",
    213: "Leguminosas – Otros",
    215: "Tubérculos – Patata (potato)",
    216: "Tubérculos – Remolacha",
    221: "Forraje – Praderas/perennes",
    223: "Forraje – Cultivos de corte",
    230: "Aceite – Colza (rapeseed)",
    231: "Aceite – Girasol",
    232: "Aceite – Otros oleaginosos",
    240: "Huerto/Frutales – Viñedo",
    244: "Huerto/Frutales – Frutales de pepita/drupe",
    250: "Otros – Superficies no agrícolas"
}

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

def evaluar_segmentacion(y_true, y_pred, n_clases, clase_a_nombre=None):
    """
    Evalúa la segmentación comparando máscaras reales y predichas.
    Muestra IoU por clase, precisión por clase y visualizaciones.
```

```
    Args:
        y_true: np.array (H, W) máscara real (valores de 0 a n_clases-1)
        y_pred: np.array (H, W) predicción (valores de 0 a n_clases-1)
        n_clases: número total de clases
        clase_a_nombre: dict opcional para mapear índice a nombre
    """

    iou_por_clase = []
    precision_por_clase = []

    print("Métricas por clase:")
    print("-" * 30)

    for clase in range(n_clases):
        interseccion = np.logical_and(y_true == clase, y_pred == clase).sum()
        union = np.logical_or(y_true == clase, y_pred == clase).sum()
        total_pred = (y_pred == clase).sum()

        iou = interseccion / union if union != 0 else 0.0
        precision = interseccion / total_pred if total_pred != 0 else 0.0

        iou_por_clase.append(iou)
        precision_por_clase.append(precision)

        nombre = clase_a_nombre.get(clase, str(clase)) if clase_a_nombre else str(clase)
        print(f"Clase {nombre}: IoU = {iou:.3f}, Precisión = {precision:.3f}")

    # Visualización por clase
    fig, axs = plt.subplots(n_clases, 3, figsize=(10, 4 * n_clases))

    for clase in range(n_clases):
        nombre = clase_a_nombre.get(clase, f"Clase {clase}") if clase_a_nombre else f"Clase {

        axs[clase, 0].imshow(y_true == clase, cmap='gray')
        axs[clase, 0].set_title(f"Máscara real - {nombre}")

        axs[clase, 1].imshow(y_pred == clase, cmap='gray')
        axs[clase, 1].set_title(f"Predicción - {nombre}")

        axs[clase, 2].imshow(np.logical_and(y_true == clase, y_pred == clase), cmap='Greens')
        axs[clase, 2].set_title(f"Aciertos - {nombre}")

        for j in range(3):
            axs[clase, j].axis('off')

    plt.tight_layout()
    plt.show()

    return iou_por_clase, precision_por_clase
```

YOLO

In [53]:

```
import rasterio
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

folder = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB/"
output_folder = os.path.join(folder, "rgb_yolo")
os.makedirs(output_folder, exist_ok=True)

for tif_file in sorted(glob(os.path.join(folder, 'AMB_*.tif'))):
```

```
    with rasterio.open(tif_file) as src:
        img = src.read([1, 2, 3])  # B4, B3, B2 (ajustar si es Sentinel)
        img = np.clip(img, 0, 10000) / 10000.0
        img = (img.transpose(1, 2, 0) * 255).astype(np.uint8)

        name = os.path.basename(tif_file).replace('.tif', '.jpg')
        Image.fromarray(img).save(os.path.join(output_folder, name))
```

In [1]:

```
import os
import xml.etree.ElementTree as ET
import json

# Carpeta base
CARPETA = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo"

def convertir_xml_a_json(path_xml):
    tree = ET.parse(path_xml)
    root = tree.getroot()

    filename = root.find("filename").text
    size = root.find("size")
    width = int(size.find("width").text)
    height = int(size.find("height").text)

    anotaciones = []
    for obj in root.findall("object"):
        label = obj.find("name").text
        bndbox = obj.find("bndbox")
        xmin = int(bndbox.find("xmin").text)
        ymin = int(bndbox.find("ymin").text)
        xmax = int(bndbox.find("xmax").text)
        ymax = int(bndbox.find("ymax").text)

        # Coordenadas centradas (como tus otros JSON)
        x_center = (xmin + xmax) / 2
        y_center = (ymin + ymax) / 2
        width_box = xmax - xmin
        height_box = ymax - ymin

        anotaciones.append({
            "label": label,
            "coordinates": {
                "x": x_center,
                "y": y_center,
                "width": width_box,
                "height": height_box
            }
        })

    json_data = [{
        "image": filename,
        "verified": False,
        "annotations": anotaciones
    }]

    # Guardar como .json
    json_path = os.path.splitext(path_xml)[0] + ".json"
    with open(json_path, "w") as f:
        json.dump(json_data, f, indent=4)

    # Eliminar el XML original
    os.remove(path_xml)
    print(f"Convertido y eliminado: {os.path.basename(path_xml)}")

# Recorre carpeta y convierte todos los .xml
```

```
for archivo in os.listdir(CARPETA):
    if archivo.endswith(".xml"):
        convertir_xml_a_json(os.path.join(CARPETA, archivo))

print("Conversión completa.")
```

```
Convertido y eliminado: AMB_R0_C1_2018.xml
Convertido y eliminado: AMB_R0_C2_2018.xml
Convertido y eliminado: AMB_R0_C3_2018.xml
Convertido y eliminado: AMB_R1_C0_2018.xml
Convertido y eliminado: AMB_R1_C1_2018.xml
Convertido y eliminado: AMB_R1_C2_2018.xml
Convertido y eliminado: AMB_R1_C3_2018.xml
Convertido y eliminado: AMB_R1_C4_2018.xml
Convertido y eliminado: AMB_R2_C0_2018.xml
Convertido y eliminado: AMB_R2_C1_2018.xml
Convertido y eliminado: AMB_R2_C2_2018.xml
Convertido y eliminado: AMB_R2_C3_2018.xml
Convertido y eliminado: AMB_R2_C4_2018.xml
Convertido y eliminado: AMB_R3_C0_2018.xml
Convertido y eliminado: AMB_R3_C1_2018.xml
Convertido y eliminado: AMB_R3_C2_2018.xml
Convertido y eliminado: AMB_R3_C3_2018.xml
Convertido y eliminado: AMB_R3_C4_2018.xml
Convertido y eliminado: AMB_R3_C5_2018.xml
Convertido y eliminado: AMB_R4_C0_2018.xml
Convertido y eliminado: AMB_R4_C1_2018.xml
Convertido y eliminado: AMB_R4_C2_2018.xml
Convertido y eliminado: AMB_R4_C3_2018.xml
Convertido y eliminado: AMB_R4_C4_2018.xml
Convertido y eliminado: AMB_R4_C5_2018.xml
Convertido y eliminado: AMB_R4_C6_2018.xml
Convertido y eliminado: AMB_R5_C0_2018.xml
Convertido y eliminado: AMB_R5_C1_2018.xml
Convertido y eliminado: AMB_R5_C2_2018.xml
Convertido y eliminado: AMB_R5_C3_2018.xml
Convertido y eliminado: AMB_R5_C4_2018.xml
Convertido y eliminado: AMB_R5_C5_2018.xml
Convertido y eliminado: AMB_R5_C6_2018.xml
Convertido y eliminado: AMB_R5_C7_2018.xml
Convertido y eliminado: AMB_R5_C8_2018.xml
Convertido y eliminado: AMB_R6_C0_2018.xml
Convertido y eliminado: AMB_R6_C1_2018.xml
Convertido y eliminado: AMB_R6_C2_2018.xml
Convertido y eliminado: AMB_R6_C3_2018.xml
Convertido y eliminado: AMB_R6_C4_2018.xml
Convertido y eliminado: AMB_R6_C5_2018.xml
Convertido y eliminado: AMB_R6_C6_2018.xml
Conversión completa.
```

In [35]:

```
import os
import json
from collections import Counter

# Ruta a la carpeta donde están los .json
CARPETA = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo"

# Contador para las clases
clases = Counter()

# Recorrer todos los archivos .json
for archivo in os.listdir(CARPETA):
    if archivo.endswith(".json"):
        ruta = os.path.join(CARPETA, archivo)
        try:
            with open(ruta, 'r') as f:
```

```
                data = json.load(f)
                if not data or "annotations" not in data[0]:
                    continue

                for ann in data[0]["annotations"]:
                    label = ann["label"]
                    clases[label] += 1
        except Exception as e:
            print(f"Error procesando {archivo}: {e}")

# Mostrar resultados
print("\nClases encontradas:")
for clase in sorted(clases):
    print(f"- {clase}: {clases[clase]} anotaciones")

print(f"\nTotal de clases distintas: {len(clases)}")
```

```
Clases encontradas:
- Agricola: 50 anotaciones
- Industria: 118 anotaciones
- Poblacion: 151 anotaciones

Total de clases distintas: 3
```

In [3]:

```
import os
import xml.etree.ElementTree as ET
import json
from PIL import Image

# Ruta base
base_dir = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo"
output_dir = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo\txt"

import os
import json

# Ruta de los .json
CARPETA_JSON = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo"

# Conjunto para clases únicas
clases_unicas = set()

# Buscar todas las clases en los .json
for archivo in os.listdir(CARPETA_JSON):
    if archivo.endswith(".json"):
        try:
            with open(os.path.join(CARPETA_JSON, archivo), "r") as f:
                data = json.load(f)
                if not data or "annotations" not in data[0]:
                    continue

                for ann in data[0]["annotations"]:
                    label = ann["label"].strip()
                    clases_unicas.add(label)
        except Exception as e:
            print(f"Error leyendo {archivo}: {e}")

# Ordenar alfabéticamente (opcional) o por aparición
clases_ordenadas = sorted(clases_unicas)  # usa list(clases_unicas) para orden por aparición

# Crear diccionario clase → ID
clase_a_id = {clase: idx for idx, clase in enumerate(clases_ordenadas)}

# Mostrar resultado
print("Diccionario generado:\n")
print("clase_a_id = {")
```

```python
for k, v in clase_a_id.items():
    print(f"    \"{k}\": {v},")
print("}")

# Función para convertir XML (PASCAL VOC)
def convertir_xml(xml_path):
    img_path = xml_path.replace(".xml", ".jpg")
    txt_path = xml_path.replace(".xml", ".txt")

    if not os.path.exists(img_path):
        print(f"Imagen no encontrada para {xml_path}")
        return

    img = Image.open(img_path)
    w, h = img.size

    tree = ET.parse(xml_path)
    root = tree.getroot()

    yolo_lines = []
    for obj in root.findall("object"):
        label = obj.find("name").text
        if label not in clase_a_id:
            continue
        clase = clase_a_id[label]

        bndbox = obj.find("bndbox")
        xmin = int(bndbox.find("xmin").text)
        ymin = int(bndbox.find("ymin").text)
        xmax = int(bndbox.find("xmax").text)
        ymax = int(bndbox.find("ymax").text)

        x_center = (xmin + xmax) / 2 / w
        y_center = (ymin + ymax) / 2 / h
        box_width = (xmax - xmin) / w
        box_height = (ymax - ymin) / h

        yolo_lines.append(f"{clase} {x_center:.6f} {y_center:.6f} {box_width:.6f} {box_height

    with open(txt_path, "w") as f:
        f.write("\n".join(yolo_lines))
    print(f"Convertido XML: {os.path.basename(txt_path)}")

# Función para convertir JSON
def convertir_json(json_path):
    img_path = json_path.replace(".json", ".jpg")
    txt_path = json_path.replace(".json", ".txt")

    if not os.path.exists(img_path):
        print(f"Imagen no encontrada para {json_path}")
        return

    img = Image.open(img_path)
    w, h = img.size

    with open(json_path, "r") as f:
        data = json.load(f)

    yolo_lines = []
    for ann in data[0].get("annotations", []):
        label = ann["label"]
        if label not in clase_a_id:
            continue
        clase = clase_a_id[label]

        coords = ann["coordinates"]
```

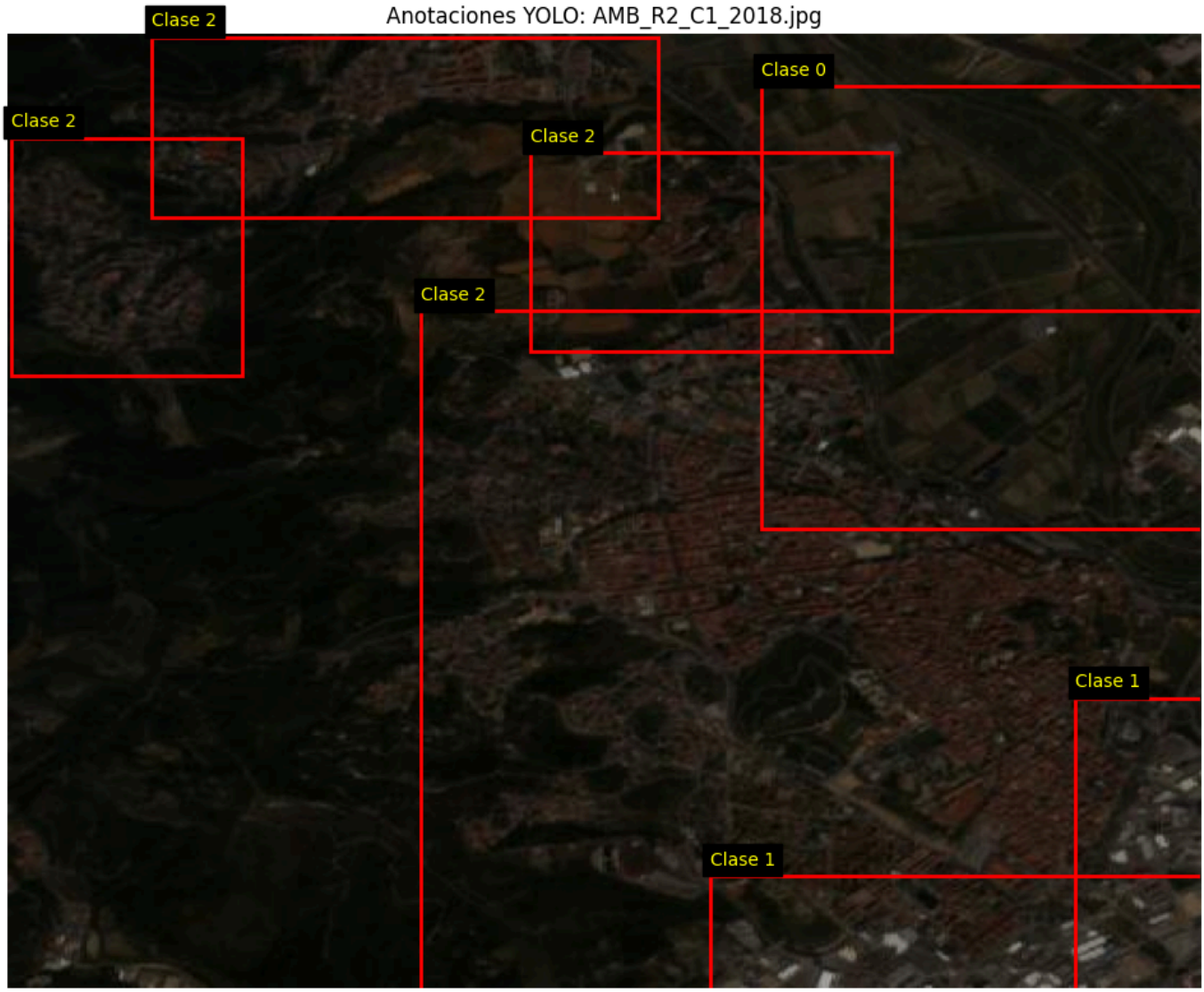
```
        x_center = coords["x"] / w
        y_center = coords["y"] / h
        width = coords["width"] / w
        height = coords["height"] / h

        yolo_lines.append(f"{clase} {x_center:.6f} {y_center:.6f} {width:.6f} {height:.6f}")

    with open(txt_path, "w") as f:
        f.write("\n".join(yolo_lines))
    print(f"Convertido JSON: {os.path.basename(txt_path)}")

# Recorremos todos los archivos de la carpeta
for file in os.listdir(base_dir):
    full_path = os.path.join(base_dir, file)

    if file.endswith(".xml"):
        convertir_xml(full_path)
    elif file.endswith(".json"):
        convertir_json(full_path)

print("Conversión completa.")
```

```
Diccionario generado:

clase_a_id = {
    "Agricola": 0,
    "Industria": 1,
    "Poblacion": 2,
}
Convertido JSON: AMB_R0_C1_2018.txt
Convertido JSON: AMB_R0_C2_2018.txt
Convertido JSON: AMB_R0_C3_2018.txt
Convertido JSON: AMB_R1_C0_2018.txt
Convertido JSON: AMB_R1_C1_2018.txt
Convertido JSON: AMB_R1_C2_2018.txt
Convertido JSON: AMB_R1_C3_2018.txt
Convertido JSON: AMB_R1_C4_2018.txt
Convertido JSON: AMB_R2_C0_2018.txt
Convertido JSON: AMB_R2_C1_2018.txt
Convertido JSON: AMB_R2_C2_2018.txt
Convertido JSON: AMB_R2_C3_2018.txt
Convertido JSON: AMB_R2_C4_2018.txt
Convertido JSON: AMB_R3_C0_2018.txt
Convertido JSON: AMB_R3_C1_2018.txt
Convertido JSON: AMB_R3_C2_2018.txt
Convertido JSON: AMB_R3_C3_2018.txt
Convertido JSON: AMB_R3_C4_2018.txt
Convertido JSON: AMB_R3_C5_2018.txt
Convertido JSON: AMB_R4_C0_2018.txt
Convertido JSON: AMB_R4_C1_2018.txt
Convertido JSON: AMB_R4_C2_2018.txt
Convertido JSON: AMB_R4_C3_2018.txt
Convertido JSON: AMB_R4_C4_2018.txt
Convertido JSON: AMB_R4_C5_2018.txt
Convertido JSON: AMB_R4_C6_2018.txt
Convertido JSON: AMB_R5_C0_2018.txt
Convertido JSON: AMB_R5_C1_2018.txt
Convertido JSON: AMB_R5_C2_2018.txt
Convertido JSON: AMB_R5_C3_2018.txt
Convertido JSON: AMB_R5_C4_2018.txt
Convertido JSON: AMB_R5_C5_2018.txt
Convertido JSON: AMB_R5_C6_2018.txt
Convertido JSON: AMB_R5_C7_2018.txt
Convertido JSON: AMB_R5_C8_2018.txt
Convertido JSON: AMB_R6_C0_2018.txt
Convertido JSON: AMB_R6_C1_2018.txt
Convertido JSON: AMB_R6_C2_2018.txt
Convertido JSON: AMB_R6_C3_2018.txt
Convertido JSON: AMB_R6_C4_2018.txt
Convertido JSON: AMB_R6_C5_2018.txt
Convertido JSON: AMB_R6_C6_2018.txt
Convertido JSON: AMB_R6_C7_2018.txt
Convertido JSON: AMB_R6_C8_2018.txt
Convertido JSON: AMB_R6_C9_2018.txt
Convertido JSON: AMB_R7_C0_2018.txt
Convertido JSON: AMB_R7_C1_2018.txt
Convertido JSON: AMB_R7_C2_2018.txt
Convertido JSON: AMB_R7_C3_2018.txt
Convertido JSON: AMB_R7_C4_2018.txt
Convertido JSON: AMB_R7_C5_2018.txt
Convertido JSON: AMB_R7_C6_2018.txt
Convertido JSON: AMB_R7_C9_2018.txt
Convertido JSON: AMB_R8_C1_2018.txt
Convertido JSON: AMB_R8_C2_2018.txt
Convertido JSON: AMB_R8_C4_2018.txt
Convertido JSON: AMB_R8_C5_2018.txt
Convertido JSON: AMB_R8_C6_2018.txt
Convertido JSON: AMB_R8_C7_2018.txt
```

```
Convertido JSON: AMB_R8_C9_2018.txt
Convertido JSON: AMB_R9_C2_2018.txt
Convertido JSON: AMB_R9_C3_2018.txt
Convertido JSON: AMB_R9_C4_2018.txt
Convertido JSON: AMB_R9_C5_2018.txt
Convertido JSON: AMB_R9_C6_2018.txt
Convertido JSON: AMB_R9_C7_2018.txt
Convertido JSON: AMB_R9_C8_2018.txt
Convertido JSON: AMB_R9_C9_2018.txt
Conversión completa.
```

In [4]:

```python
import os
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from PIL import Image

# Ruta base
IMG_NAME = "AMB_R2_C1_2018.jpg"  # Cambia por el que quieras
IMG_DIR = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo"
TXT_DIR = IMG_DIR  # Si los .txt están en el mismo sitio

# Leer imagen
img_path = os.path.join(IMG_DIR, IMG_NAME)
img = Image.open(img_path)
w, h = img.size

# Leer anotaciones YOLO
txt_name = os.path.splitext(IMG_NAME)[0] + ".txt"
txt_path = os.path.join(TXT_DIR, txt_name)

boxes = []
with open(txt_path, "r") as f:
    for line in f:
        clase_id, x, y, w_box, h_box = map(float, line.strip().split())
        boxes.append((clase_id, x, y, w_box, h_box))

# Mostrar imagen
fig, ax = plt.subplots(1, figsize=(10, 8))
ax.imshow(img)

# Dibujar cada caja
for clase_id, x, y, w_box, h_box in boxes:
    # Convertir a coordenadas absolutas
    x_abs = (x - w_box / 2) * w
    y_abs = (y - h_box / 2) * h
    width = w_box * w
    height = h_box * h

    rect = patches.Rectangle(
        (x_abs, y_abs),
        width,
        height,
        linewidth=2,
        edgecolor='red',
        facecolor='none'
    )
    ax.add_patch(rect)
    ax.text(x_abs, y_abs - 5, f"Clase {int(clase_id)}", color='yellow', fontsize=10, backgrou

ax.set_title(f"Anotaciones YOLO: {IMG_NAME}")
plt.axis('off')
plt.tight_layout()
plt.show()
```



In [6]:
```python
import os
import shutil
import random

def dividir_desde_raiz(carpeta_raiz, carpeta_train, carpeta_val, val_ratio=0.2):
    # Crear carpetas destino
    for base in [carpeta_train, carpeta_val]:
        os.makedirs(os.path.join(base, "images"), exist_ok=True)
        os.makedirs(os.path.join(base, "labels"), exist_ok=True)

    # Buscar imágenes con .txt
    archivos = os.listdir(carpeta_raiz)
    imagenes = [
        f for f in archivos
        if f.lower().endswith((".jpg", ".jpeg", ".png"))
        and os.path.exists(os.path.join(carpeta_raiz, f.rsplit(".", 1)[0] + ".txt"))
    ]

    print(f"Encontradas {len(imagenes)} imágenes con sus .txt")

    random.shuffle(imagenes)
    val_size = int(len(imagenes) * val_ratio)

    for i, img in enumerate(imagenes):
        base_name = os.path.splitext(img)[0]
        label_file = base_name + ".txt"

        destino = carpeta_val if i < val_size else carpeta_train

        # Copiar imagen
        shutil.copy(os.path.join(carpeta_raiz, img), os.path.join(destino, "images", img))
```

```
        # Copiar etiqueta
        shutil.copy(os.path.join(carpeta_raiz, label_file), os.path.join(destino, "labels", l

    print(f"Dataset dividido: {len(imagenes) - val_size} entrenamiento, {val_size} validación

# 📂 Carpetas
CARPETA_RAIZ = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo"
CARPETA_TRAIN = os.path.join(CARPETA_RAIZ, "train")
CARPETA_VAL = os.path.join(CARPETA_RAIZ, "val")

# Ejecutar
dividir_desde_raiz(CARPETA_RAIZ, CARPETA_TRAIN, CARPETA_VAL)
```

```
Encontradas 68 imágenes con sus .txt
Dataset dividido: 55 entrenamiento, 13 validación
```

In [2]:

```
import os
os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"
from ultralytics import YOLO

def entrenar_yolo(data_path, modelo_salida, epochs=1000, img_size=640):
    """
    Entrena un modelo YOLOv8 con el dataset proporcionado.
    """
    modelo = YOLO("yolov8n.pt")
    modelo.train(
        data=data_path,
        epochs=epochs,
        imgsz=img_size,
        save_period=10,
        project=os.path.dirname(modelo_salida),
        name=os.path.basename(modelo_salida).split('.')[0]
    )

DATA_PATH = r"data_other_pc.yaml"
MODELO_SALIDA = r"yolov8_model.pt"
entrenar_yolo(DATA_PATH, MODELO_SALIDA,100)
```

```
New https://pypi.org/project/ultralytics/8.3.155 available  Update with 'pip install -U ultral
ytics'
Ultralytics 8.3.39  Python-3.10.4 torch-2.1.0+cpu CPU (Intel Core(TM) i7-10750H 2.60GHz)
engine\trainer: task=detect, mode=train, model=yolov8n.pt, data=data_other_pc.yaml, epochs=10
0, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=10, cache=False, devic
e=None, workers=8, project=, name=yolov8_model9, exist_ok=False, pretrained=True, optimizer=au
to, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, clos
e_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi_scale=Fal
se, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_h
ybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None,
vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=False, classes
=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_con
f=False, save_crop=False, show_labels=True, show_conf=True, show_boxes=True, line_width=None,
format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=True, ops
et=None, workspace=None, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, w
armup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.
0, kobj=1.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5,
shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste
=0.0, copy_paste_mode=flip, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=Non
e, tracker=botsort.yaml, save_dir=runs\detect\yolov8_model9
Overriding model.yaml nc=80 with nc=3

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3,
2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3,
2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1,
True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3,
2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2,
True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128,
3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128,
2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256,
3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256,
1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256,
5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2,
'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128,
1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2,
'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64,
1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3,
2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128,
1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128,
3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256,
1]
 22        [15, 18, 21]  1    751897  ultralytics.nn.modules.head.Detect           [3, [64, 12
8, 256]]
Model summary: 225 layers, 3,011,433 parameters, 3,011,417 gradients, 8.2 GFLOPs
```

```
Transferred 319/355 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs\detect\yolov8_model9', view at http://local
host:6006/
Freezing layer 'model.22.dfl.conv.weight'
train: Scanning C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo\train\labels.cache... 55 im
ages, 2 backgrounds, 0 corrupt: 100%|██████████| 55/55 [00:00<?, ?it/s]
val: Scanning C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo\val\labels.cache... 13 image
s, 0 backgrounds, 0 corrupt: 100%|██████████| 13/13 [00:00<?, ?it/s]
Plotting labels to runs\detect\yolov8_model9\labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining be
st 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.001429, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 wei
ght(decay=0.0005), 63 bias(decay=0.0)
TensorBoard: model graph visualization added
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\detect\yolov8_model9
Starting training for 100 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/100         0G      2.314      3.523       2.31         55        640: 100%|██████████
| 4/4 [00:14<00:00,  3.60s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.62s/it]
                   all         13         53    0.00951      0.663     0.0405     0.0179

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/100         0G      2.084      3.434      2.162         68        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.40s/it]
                   all         13         53    0.00982      0.674     0.0546      0.022

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/100         0G      1.942      3.359      2.061         60        640: 100%|██████████
| 4/4 [00:13<00:00,  3.33s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.44s/it]
                   all         13         53    0.00905      0.618     0.0819     0.0333

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      4/100         0G      1.938      3.296      2.033         75        640: 100%|██████████
| 4/4 [00:13<00:00,  3.36s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.25s/it]
                   all         13         53    0.00959      0.679     0.0946     0.0394

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      5/100         0G      1.948      3.123      2.057         51        640: 100%|██████████
| 4/4 [00:13<00:00,  3.32s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.20s/it]
                   all         13         53     0.0102      0.709     0.126     0.0473

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      6/100         0G      1.817      2.954      1.977         66        640: 100%|██████████
| 4/4 [00:13<00:00,  3.26s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.29s/it]
                   all         13         53    0.00949      0.648     0.115     0.0413

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      7/100         0G      1.831      2.934      1.956         67        640: 100%|██████████
| 4/4 [00:13<00:00,  3.29s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.26s/it]
```

```
                   all         13         53    0.00953      0.629      0.094     0.0325

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

      8/100         0G      1.896      2.779      1.978         72        640: 100%|██████████
| 4/4 [00:13<00:00,  3.32s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53     0.0094      0.627      0.116     0.0352

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

      9/100         0G      1.779       2.69      1.898         67        640: 100%|██████████
| 4/4 [00:13<00:00,  3.27s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.17s/it]
                   all         13         53      0.721     0.0533      0.124      0.042

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     10/100         0G      1.812      2.556       1.91         56        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.729     0.0933     0.0674     0.0204

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     11/100         0G      1.756      2.584      1.895         86        640: 100%|██████████
| 4/4 [00:13<00:00,  3.41s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.529      0.136      0.122     0.0382

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     12/100         0G      1.765      2.449      1.893         64        640: 100%|██████████
| 4/4 [00:13<00:00,  3.33s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.541      0.138      0.136     0.0404

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     13/100         0G       1.78      2.389      1.836         70        640: 100%|██████████
| 4/4 [00:13<00:00,  3.33s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.462      0.199      0.127     0.0435

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     14/100         0G      1.848      2.494      1.917         48        640: 100%|██████████
| 4/4 [00:13<00:00,  3.26s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.486      0.232       0.12     0.0435

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     15/100         0G      1.708      2.334      1.795         44        640: 100%|██████████
| 4/4 [00:13<00:00,  3.26s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.16s/it]
                   all         13         53      0.515      0.214      0.112     0.0392

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     16/100         0G      1.745      2.272      1.802         84        640: 100%|██████████
| 4/4 [00:13<00:00,  3.36s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.02s/it]
                   all         13         53      0.549      0.277      0.154     0.0567

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     17/100         0G      1.732      2.285      1.821         55        640: 100%|██████████
| 4/4 [00:13<00:00,  3.32s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.06s/it]
```

```
                   all         13         53      0.559      0.248      0.191     0.0681

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     18/100         0G      1.692      2.224      1.768         63        640: 100%|██████████
| 4/4 [00:13<00:00,  3.39s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.09s/it]
                   all         13         53      0.551      0.229      0.234     0.0816

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     19/100         0G      1.657      2.123      1.769        101        640: 100%|██████████
| 4/4 [00:13<00:00,  3.28s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.20s/it]
                   all         13         53      0.521      0.189      0.207     0.0708

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     20/100         0G      1.613      2.105      1.756         60        640: 100%|██████████
| 4/4 [00:13<00:00,  3.34s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.04s/it]
                   all         13         53      0.738      0.133      0.185     0.0677

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     21/100         0G      1.568       2.08      1.728         54        640: 100%|██████████
| 4/4 [00:13<00:00,  3.26s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.18s/it]
                   all         13         53      0.538      0.207      0.201     0.0562

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     22/100         0G      1.654      2.127      1.784         82        640: 100%|██████████
| 4/4 [00:13<00:00,  3.28s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.20s/it]
                   all         13         53       0.55      0.316      0.269     0.0717

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     23/100         0G      1.614      2.069      1.735         43        640: 100%|██████████
| 4/4 [00:13<00:00,  3.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.231       0.46      0.245     0.0717

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     24/100         0G      1.589      2.065      1.733         44        640: 100%|██████████
| 4/4 [00:13<00:00,  3.42s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53       0.21      0.446      0.203     0.0662

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     25/100         0G       1.58      1.956      1.725         49        640: 100%|██████████
| 4/4 [00:13<00:00,  3.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.14s/it]
                   all         13         53       0.35      0.331      0.193     0.0588

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     26/100         0G      1.464      1.978      1.623         54        640: 100%|██████████
| 4/4 [00:13<00:00,  3.39s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.306      0.373      0.201     0.0651

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     27/100         0G      1.523       1.98      1.672         36        640: 100%|██████████
| 4/4 [00:13<00:00,  3.38s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.11s/it]
```

```
                   all         13         53      0.366      0.356      0.219     0.0738

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     28/100         0G      1.609      1.948      1.728         66        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.397      0.465      0.281      0.118

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     29/100         0G      1.616      1.967      1.749         44        640: 100%|██████████
| 4/4 [00:13<00:00,  3.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.15s/it]
                   all         13         53      0.182        0.5      0.224     0.0878

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     30/100         0G      1.559      1.832      1.684         70        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.15s/it]
                   all         13         53      0.233      0.438       0.17     0.0708

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     31/100         0G      1.479      1.804      1.604         52        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.459       0.25      0.183     0.0716

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     32/100         0G      1.529      1.814      1.625         57        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.527      0.167      0.214     0.0765

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     33/100         0G      1.506       1.76      1.623         61        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.15s/it]
                   all         13         53      0.454      0.236      0.151     0.0625

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     34/100         0G      1.457      1.706      1.582         44        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.18s/it]
                   all         13         53      0.512      0.191      0.146     0.0611

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     35/100         0G      1.505      1.738      1.609         82        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.16s/it]
                   all         13         53       0.17      0.401      0.167     0.0639

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     36/100         0G       1.48      1.688       1.58         77        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.232       0.39       0.17     0.0613

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     37/100         0G      1.434      1.744      1.587         52        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
```

```
                   all         13         53      0.208      0.369      0.238     0.0753

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     38/100         0G      1.415      1.694      1.557         60        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53       0.26      0.467      0.274     0.0911

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     39/100         0G      1.395      1.625      1.532         51        640: 100%|██████████
| 4/4 [00:14<00:00,  3.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.23s/it]
                   all         13         53      0.286      0.592      0.285     0.0991

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     40/100         0G      1.431      1.711      1.591         57        640: 100%|██████████
| 4/4 [00:13<00:00,  3.50s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.219      0.467      0.218     0.0737

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     41/100         0G       1.36      1.544       1.51         69        640: 100%|██████████
| 4/4 [00:13<00:00,  3.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.266      0.366      0.214     0.0664

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     42/100         0G      1.351      1.526      1.508         71        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53       0.27      0.292      0.182     0.0635

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     43/100         0G       1.31      1.426      1.459         63        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.241      0.286      0.163     0.0603

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     44/100         0G      1.346      1.482      1.473         83        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.11s/it]
                   all         13         53       0.27      0.308      0.197     0.0685

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     45/100         0G      1.368      1.527      1.501         56        640: 100%|██████████
| 4/4 [00:14<00:00,  3.57s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.18s/it]
                   all         13         53      0.305      0.366      0.234     0.0724

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     46/100         0G      1.317      1.467      1.448         86        640: 100%|██████████
| 4/4 [00:14<00:00,  3.57s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.225      0.377      0.176     0.0557

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     47/100         0G      1.337      1.548      1.435         49        640: 100%|██████████
| 4/4 [00:13<00:00,  3.41s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
```

```
                   all         13         53      0.244      0.229      0.176     0.0518

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     48/100         0G      1.267       1.38      1.441         54        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.278      0.316      0.179     0.0582

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     49/100         0G      1.277      1.452      1.417         76        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.11s/it]
                   all         13         53      0.232      0.388      0.227     0.0751

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     50/100         0G      1.161      1.327      1.358         65        640: 100%|██████████
| 4/4 [00:14<00:00,  3.61s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.584      0.282      0.241     0.0852

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     51/100         0G      1.324      1.493      1.463         63        640: 100%|██████████
| 4/4 [00:13<00:00,  3.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.04s/it]
                   all         13         53      0.395      0.349      0.267      0.103

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     52/100         0G      1.242       1.34      1.414         69        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.508       0.34      0.287      0.102

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     53/100         0G      1.292      1.458      1.468         71        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.322      0.431       0.29       0.11

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     54/100         0G      1.311      1.505      1.493         42        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.298      0.321      0.244     0.0997

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     55/100         0G      1.224      1.483      1.393         37        640: 100%|██████████
| 4/4 [00:14<00:00,  3.63s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53       0.26      0.459      0.265      0.107

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     56/100         0G      1.173      1.321      1.362         74        640: 100%|██████████
| 4/4 [00:13<00:00,  3.49s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.06s/it]
                   all         13         53       0.31      0.467      0.311      0.106

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     57/100         0G      1.213      1.308      1.392         55        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.06s/it]
```

```
                   all         13         53      0.344      0.428      0.319     0.0895

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     58/100         0G        1.2      1.283      1.323         83        640: 100%|██████████
| 4/4 [00:13<00:00,  3.46s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.06s/it]
                   all         13         53      0.217      0.388      0.221     0.0759

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     59/100         0G      1.184      1.314      1.354         67        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.231      0.397      0.207     0.0766

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     60/100         0G      1.186      1.279      1.368         87        640: 100%|██████████
| 4/4 [00:14<00:00,  3.61s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.212      0.399      0.213     0.0788

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     61/100         0G      1.138       1.17      1.281         91        640: 100%|██████████
| 4/4 [00:14<00:00,  3.52s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53       0.29      0.372      0.223       0.09

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     62/100         0G      1.154      1.257      1.363         85        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.469      0.305      0.264      0.111

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     63/100         0G       1.13      1.204      1.329         80        640: 100%|██████████
| 4/4 [00:13<00:00,  3.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.06s/it]
                   all         13         53      0.513      0.305      0.286      0.116

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     64/100         0G      1.063      1.098      1.261         65        640: 100%|██████████
| 4/4 [00:14<00:00,  3.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53       0.37      0.393      0.298      0.119

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     65/100         0G      1.035      1.115      1.243         63        640: 100%|██████████
| 4/4 [00:14<00:00,  3.60s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.367      0.366      0.283      0.105

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     66/100         0G      1.169      1.399      1.377         44        640: 100%|██████████
| 4/4 [00:13<00:00,  3.41s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.477       0.26      0.236     0.0884

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     67/100         0G      1.153      1.188      1.363         52        640: 100%|██████████
| 4/4 [00:13<00:00,  3.42s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
```

```
                   all         13         53      0.193      0.274      0.165     0.0654

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     68/100         0G       1.14      1.257      1.343         59        640: 100%|██████████
| 4/4 [00:13<00:00,  3.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.577      0.215      0.187     0.0621

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     69/100         0G      1.085      1.143      1.308         62        640: 100%|██████████
| 4/4 [00:14<00:00,  3.60s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.11s/it]
                   all         13         53      0.178      0.229      0.172     0.0594

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     70/100         0G      1.205      1.404      1.369         60        640: 100%|██████████
| 4/4 [00:14<00:00,  3.71s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.219      0.276      0.176     0.0631

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     71/100         0G      1.109      1.197      1.285         84        640: 100%|██████████
| 4/4 [00:13<00:00,  3.48s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.179      0.247      0.178      0.063

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     72/100         0G       1.03      1.124      1.255         73        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.201      0.313       0.18     0.0599

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     73/100         0G       1.11      1.182      1.327         65        640: 100%|██████████
| 4/4 [00:14<00:00,  3.52s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.17s/it]
                   all         13         53      0.212      0.358      0.195     0.0638

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     74/100         0G      1.026       1.07      1.223         52        640: 100%|██████████
| 4/4 [00:14<00:00,  3.58s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53       0.28       0.33      0.233     0.0728

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     75/100         0G     0.9904      1.041      1.225         83        640: 100%|██████████
| 4/4 [00:13<00:00,  3.45s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.232       0.37      0.223      0.077

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     76/100         0G      1.112      1.169      1.316         60        640: 100%|██████████
| 4/4 [00:13<00:00,  3.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.174      0.444      0.219     0.0807

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     77/100         0G      1.018      1.098       1.26         83        640: 100%|██████████
| 4/4 [00:14<00:00,  3.61s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.18s/it]
```

```
                   all         13         53      0.294      0.307      0.227     0.0858

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     78/100         0G      1.068      1.072      1.261         83        640: 100%|██████████
| 4/4 [00:14<00:00,  3.58s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.241      0.338      0.233     0.0871

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     79/100         0G     0.9921      1.104      1.231         46        640: 100%|██████████
| 4/4 [00:13<00:00,  3.42s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.10s/it]
                   all         13         53      0.189      0.291      0.212     0.0827

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     80/100         0G     0.9841      1.081      1.208         52        640: 100%|██████████
| 4/4 [00:13<00:00,  3.48s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.198      0.294      0.201     0.0812

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     81/100         0G     0.9562      1.005        1.2         89        640: 100%|██████████
| 4/4 [00:14<00:00,  3.53s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.15s/it]
                   all         13         53      0.186      0.336      0.208     0.0742

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     82/100         0G      1.006      1.077      1.256         66        640: 100%|██████████
| 4/4 [00:14<00:00,  3.59s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.221      0.298      0.214     0.0749

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     83/100         0G     0.9873      1.068      1.203         69        640: 100%|██████████
| 4/4 [00:13<00:00,  3.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.229      0.303      0.206     0.0733

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     84/100         0G     0.9522      1.023       1.19         55        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.245       0.35      0.203     0.0708

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     85/100         0G     0.9121      1.014      1.179         77        640: 100%|██████████
| 4/4 [00:14<00:00,  3.52s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.264      0.343      0.203     0.0687

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     86/100         0G     0.9288     0.9923      1.178         82        640: 100%|██████████
| 4/4 [00:14<00:00,  3.60s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.266      0.359      0.197     0.0671

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     87/100         0G     0.9321     0.9776      1.166         72        640: 100%|██████████
| 4/4 [00:14<00:00,  3.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.05s/it]
```

```
                   all         13         53      0.204      0.328      0.182     0.0708

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     88/100         0G     0.9129     0.9823      1.194         68        640: 100%|██████████
| 4/4 [00:13<00:00,  3.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.223      0.351       0.19     0.0707

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     89/100         0G     0.9258     0.9919      1.208         65        640: 100%|██████████
| 4/4 [00:13<00:00,  3.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.269       0.33      0.195     0.0691

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     90/100         0G      0.873     0.9037      1.152         78        640: 100%|██████████
| 4/4 [00:14<00:00,  3.57s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.278       0.33      0.199      0.065

Closing dataloader mosaic

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     91/100         0G     0.9627      1.285      1.203         24        640: 100%|██████████
| 4/4 [00:14<00:00,  3.60s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all         13         53      0.226      0.357      0.181     0.0632

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     92/100         0G     0.9295      1.164      1.188         40        640: 100%|██████████
| 4/4 [00:13<00:00,  3.42s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.05s/it]
                   all         13         53      0.204      0.359      0.185     0.0626

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     93/100         0G     0.9343      1.113      1.226         29        640: 100%|██████████
| 4/4 [00:13<00:00,  3.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all         13         53      0.234      0.372      0.207     0.0669

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     94/100         0G     0.8767      1.035      1.173         44        640: 100%|██████████
| 4/4 [00:13<00:00,  3.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.15s/it]
                   all         13         53      0.244      0.385      0.217     0.0712

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     95/100         0G      0.822       0.99      1.151         27        640: 100%|██████████
| 4/4 [00:14<00:00,  3.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.258      0.391      0.229     0.0749

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

     96/100         0G     0.8302     0.9585      1.141         37        640: 100%|██████████
| 4/4 [00:14<00:00,  3.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.262      0.395      0.227     0.0765

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

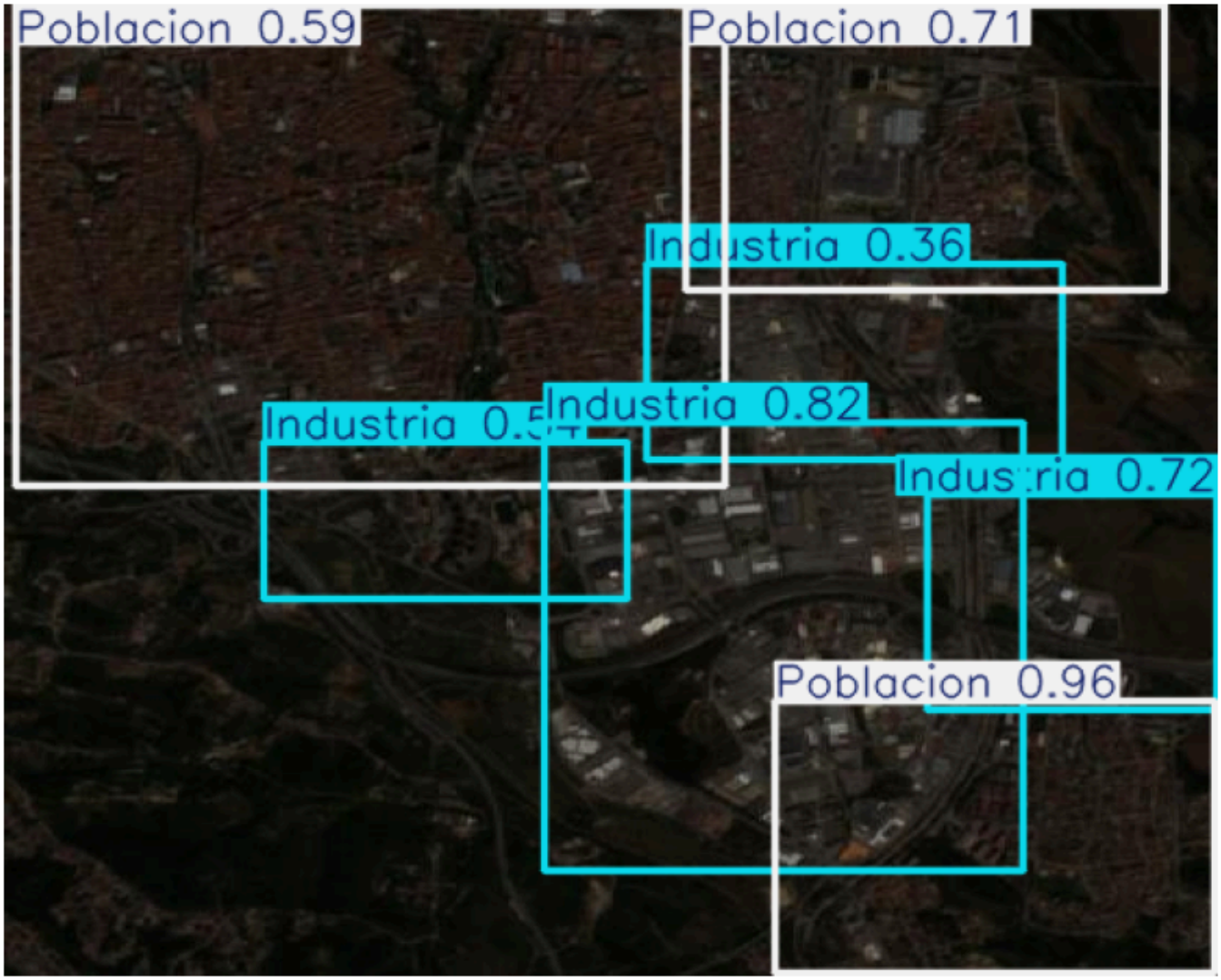
```
     97/100         0G     0.8703     0.9973      1.122         33        640: 100%|██████████
| 4/4 [00:13<00:00,  3.37s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53      0.285      0.427      0.229     0.0769

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     98/100         0G     0.8443     0.9629      1.114         37        640: 100%|██████████
| 4/4 [00:13<00:00,  3.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.07s/it]
                   all         13         53       0.28      0.433      0.245     0.0806

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     99/100         0G     0.8926       1.13      1.184         39        640: 100%|██████████
| 4/4 [00:13<00:00,  3.42s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.12s/it]
                   all         13         53      0.259      0.385      0.231     0.0777

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    100/100         0G     0.7562     0.8628      1.058         30        640: 100%|██████████
| 4/4 [00:14<00:00,  3.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:01<00:00,  1.14s/it]
                   all         13         53      0.283      0.376      0.238     0.0779

100 epochs completed in 0.426 hours.
Optimizer stripped from runs\detect\yolov8_model9\weights\last.pt, 6.2MB
Optimizer stripped from runs\detect\yolov8_model9\weights\best.pt, 6.2MB

Validating runs\detect\yolov8_model9\weights\best.pt...
Ultralytics 8.3.39  Python-3.10.4 torch-2.1.0+cpu CPU (Intel Core(TM) i7-10750H 2.60GHz)
Model summary (fused): 168 layers, 3,006,233 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
|██████████| 1/1 [00:00<00:00,  1.04it/s]
                   all         13         53      0.369      0.393      0.299       0.12
              Agricola          3          7       0.38      0.286      0.306      0.105
             Industria         10         21      0.297      0.333      0.192     0.0785
             Poblacion         12         25      0.431       0.56      0.399      0.175
              Agricola          3          7       0.38      0.286      0.306      0.105
             Industria         10         21      0.297      0.333      0.192     0.0785
             Poblacion         12         25      0.431       0.56      0.399      0.175
Speed: 1.7ms preprocess, 57.2ms inference, 0.0ms loss, 7.0ms postprocess per image
Results saved to runs\detect\yolov8_model9
```

In [6]:
```python
# Cargar el modelo entrenado
modelo = YOLO(r"C:\Users\crome\Desktop\VIU\Code\runs\detect\yolov8_model9\weights\best.pt")
```

In [8]:
```python
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# Ruta a una imagen de prueba
imagen_prueba = r"C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo\AMB_R7_C1_2018.jpg"

# Realizar la predicción
resultados = modelo(imagen_prueba)

# Dibujar las predicciones sobre la imagen
imagen_con_predicciones = resultados[0].plot()
%matplotlib inline
# Mostrar la imagen con predicciones
plt.figure(figsize=(10, 10))
plt.imshow(cv2.cvtColor(imagen_con_predicciones, cv2.COLOR_BGR2RGB))
```

```
plt.axis("off")
plt.show()
```

```
image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo\AMB_R7_C1_2018.jpg: 544x640 4 Industrias, 3 Poblacions, 115.4ms
Speed: 0.0ms preprocess, 115.4ms inference, 2.0ms postprocess per image at shape (1, 3, 544, 640)
```



In [9]:
```
# Validar el modelo en el conjunto de validación
modelo.val(data=r"data_other_pc.yaml")
```

```
Ultralytics 8.3.39  Python-3.10.4 torch-2.1.0+cpu CPU (Intel Core(TM) i7-10750H 2.60GHz)
val: Scanning C:\Users\crome\Desktop\VIU\Code\Datos\AMB\rgb_yolo\val\labels.cache... 13 images, 0 backgrounds, 0 corrupt: 100%|██████████| 13/13 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  1.10it/s]
                   all         13         53      0.369      0.393      0.299       0.12
              Agricola          3          7       0.38      0.286      0.306      0.105
             Industria         10         21      0.297      0.333      0.192     0.0785
             Poblacion         12         25      0.431       0.56      0.399      0.175
Speed: 1.7ms preprocess, 55.1ms inference, 0.0ms loss, 5.9ms postprocess per image
Results saved to runs\detect\val5
```

Out[9]: ultralytics.utils.metrics.DetMetrics object with attributes:

```
ap_class_index: array([0, 1, 2])
box: ultralytics.utils.metrics.Metric object
confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix object at 0x00000275AAB58580>
curves: ['Precision-Recall(B)', 'F1-Confidence(B)', 'Precision-Confidence(B)', 'Recall-Confid
ence(B)']
curves_results: [[array([          0,    0.001001,    0.002002,    0.003003,    0.004004,
0.005005,    0.006006,    0.007007,    0.008008,    0.009009,     0.01001,    0.011011,    0.
012012,    0.013013,    0.014014,    0.015015,    0.016016,    0.017017,    0.018018,    0.01
9019,     0.02002,    0.021021,    0.022022,    0.023023,
          0.024024,    0.025025,    0.026026,    0.027027,    0.028028,    0.029029,     0.03
003,    0.031031,    0.032032,    0.033033,    0.034034,    0.035035,    0.036036,    0.03703
7,    0.038038,    0.039039,     0.04004,    0.041041,    0.042042,    0.043043,    0.044044,
0.045045,    0.046046,    0.047047,
          0.048048,    0.049049,     0.05005,    0.051051,    0.052052,    0.053053,    0.054
054,    0.055055,    0.056056,    0.057057,    0.058058,    0.059059,     0.06006,    0.06106
1,    0.062062,    0.063063,    0.064064,    0.065065,    0.066066,    0.067067,    0.068068,
0.069069,     0.07007,    0.071071,
          0.072072,    0.073073,    0.074074,    0.075075,    0.076076,    0.077077,    0.078
078,    0.079079,     0.08008,    0.081081,    0.082082,    0.083083,    0.084084,    0.08508
5,    0.086086,    0.087087,    0.088088,    0.089089,     0.09009,    0.091091,    0.092092,
0.093093,    0.094094,    0.095095,
          0.096096,    0.097097,    0.098098,    0.099099,      0.1001,      0.1011,      0.1
021,      0.1031,      0.1041,     0.10511,     0.10611,     0.10711,     0.10811,     0.1091
1,     0.11011,     0.11111,     0.11211,     0.11311,     0.11411,     0.11512,     0.11612,
0.11712,     0.11812,     0.11912,
           0.12012,     0.12112,     0.12212,     0.12312,     0.12412,     0.12513,     0.12
613,     0.12713,     0.12813,     0.12913,     0.13013,     0.13113,     0.13213,     0.1331
3,     0.13413,     0.13514,     0.13614,     0.13714,     0.13814,     0.13914,     0.14014,
0.14114,     0.14214,     0.14314,
           0.14414,     0.14515,     0.14615,     0.14715,     0.14815,     0.14915,     0.15
015,     0.15115,     0.15215,     0.15315,     0.15415,     0.15516,     0.15616,     0.1571
6,     0.15816,     0.15916,     0.16016,     0.16116,     0.16216,     0.16316,     0.16416,
0.16517,     0.16617,     0.16717,
           0.16817,     0.16917,     0.17017,     0.17117,     0.17217,     0.17317,     0.17
417,     0.17518,     0.17618,     0.17718,     0.17818,     0.17918,     0.18018,     0.1811
8,     0.18218,     0.18318,     0.18418,     0.18519,     0.18619,     0.18719,     0.18819,
0.18919,     0.19019,     0.19119,
           0.19219,     0.19319,     0.19419,      0.1952,      0.1962,      0.1972,      0.1
982,      0.1992,      0.2002,      0.2012,      0.2022,      0.2032,      0.2042,     0.2052
1,     0.20621,     0.20721,     0.20821,     0.20921,     0.21021,     0.21121,     0.21221,
0.21321,     0.21421,     0.21522,
           0.21622,     0.21722,     0.21822,     0.21922,     0.22022,     0.22122,     0.22
222,     0.22322,     0.22422,     0.22523,     0.22623,     0.22723,     0.22823,     0.2292
3,     0.23023,     0.23123,     0.23223,     0.23323,     0.23423,     0.23524,     0.23624,
0.23724,     0.23824,     0.23924,
           0.24024,     0.24124,     0.24224,     0.24324,     0.24424,     0.24525,     0.24
625,     0.24725,     0.24825,     0.24925,     0.25025,     0.25125,     0.25225,     0.2532
5,     0.25425,     0.25526,     0.25626,     0.25726,     0.25826,     0.25926,     0.26026,
0.26126,     0.26226,     0.26326,
           0.26426,     0.26527,     0.26627,     0.26727,     0.26827,     0.26927,     0.27
027,     0.27127,     0.27227,     0.27327,     0.27427,     0.27528,     0.27628,     0.2772
8,     0.27828,     0.27928,     0.28028,     0.28128,     0.28228,     0.28328,     0.28428,
0.28529,     0.28629,     0.28729,
           0.28829,     0.28929,     0.29029,     0.29129,     0.29229,     0.29329,     0.29
429,      0.2953,      0.2963,      0.2973,      0.2983,      0.2993,      0.3003,      0.301
3,      0.3023,      0.3033,      0.3043,     0.30531,     0.30631,     0.30731,     0.30831,
0.30931,     0.31031,     0.31131,
           0.31231,     0.31331,     0.31431,     0.31532,     0.31632,     0.31732,     0.31
832,     0.31932,     0.32032,     0.32132,     0.32232,     0.32332,     0.32432,     0.3253
3,     0.32633,     0.32733,     0.32833,     0.32933,     0.33033,     0.33133,     0.33233,
0.33333,     0.33433,     0.33534,
           0.33634,     0.33734,     0.33834,     0.33934,     0.34034,     0.34134,     0.34
234,     0.34334,     0.34434,     0.34535,     0.34635,     0.34735,     0.34835,     0.3493
```

5,   0.35035,   0.35135,   0.35235,   0.35335,   0.35435,   0.35536,   0.35636,
0.35736,   0.35836,   0.35936,
          0.36036,   0.36136,   0.36236,   0.36336,   0.36436,   0.36537,   0.36
637,   0.36737,   0.36837,   0.36937,   0.37037,   0.37137,   0.37237,   0.3733
7,   0.37437,   0.37538,   0.37638,   0.37738,   0.37838,   0.37938,   0.38038,
0.38138,   0.38238,   0.38338,
          0.38438,   0.38539,   0.38639,   0.38739,   0.38839,   0.38939,   0.39
039,   0.39139,   0.39239,   0.39339,   0.39439,   0.3954,   0.3964,   0.397
4,   0.3984,   0.3994,   0.4004,   0.4014,   0.4024,   0.4034,   0.4044,
0.40541,   0.40641,   0.40741,
          0.40841,   0.40941,   0.41041,   0.41141,   0.41241,   0.41341,   0.41
441,   0.41542,   0.41642,   0.41742,   0.41842,   0.41942,   0.42042,   0.4214
2,   0.42242,   0.42342,   0.42442,   0.42543,   0.42643,   0.42743,   0.42843,
0.42943,   0.43043,   0.43143,
          0.43243,   0.43343,   0.43443,   0.43544,   0.43644,   0.43744,   0.43
844,   0.43944,   0.44044,   0.44144,   0.44244,   0.44344,   0.44444,   0.4454
5,   0.44645,   0.44745,   0.44845,   0.44945,   0.45045,   0.45145,   0.45245,
0.45345,   0.45445,   0.45546,
          0.45646,   0.45746,   0.45846,   0.45946,   0.46046,   0.46146,   0.46
246,   0.46346,   0.46446,   0.46547,   0.46647,   0.46747,   0.46847,   0.4694
7,   0.47047,   0.47147,   0.47247,   0.47347,   0.47447,   0.47548,   0.47648,
0.47748,   0.47848,   0.47948,
          0.48048,   0.48148,   0.48248,   0.48348,   0.48448,   0.48549,   0.48
649,   0.48749,   0.48849,   0.48949,   0.49049,   0.49149,   0.49249,   0.4934
9,   0.49449,   0.4955,   0.4965,   0.4975,   0.4985,   0.4995,   0.5005,
0.5015,   0.5025,   0.5035,
           0.5045,   0.50551,   0.50651,   0.50751,   0.50851,   0.50951,   0.51
051,   0.51151,   0.51251,   0.51351,   0.51451,   0.51552,   0.51652,   0.5175
2,   0.51852,   0.51952,   0.52052,   0.52152,   0.52252,   0.52352,   0.52452,
0.52553,   0.52653,   0.52753,
          0.52853,   0.52953,   0.53053,   0.53153,   0.53253,   0.53353,   0.53
453,   0.53554,   0.53654,   0.53754,   0.53854,   0.53954,   0.54054,   0.5415
4,   0.54254,   0.54354,   0.54454,   0.54555,   0.54655,   0.54755,   0.54855,
0.54955,   0.55055,   0.55155,
          0.55255,   0.55355,   0.55455,   0.55556,   0.55656,   0.55756,   0.55
856,   0.55956,   0.56056,   0.56156,   0.56256,   0.56356,   0.56456,   0.5655
7,   0.56657,   0.56757,   0.56857,   0.56957,   0.57057,   0.57157,   0.57257,
0.57357,   0.57457,   0.57558,
          0.57658,   0.57758,   0.57858,   0.57958,   0.58058,   0.58158,   0.58
258,   0.58358,   0.58458,   0.58559,   0.58659,   0.58759,   0.58859,   0.5895
9,   0.59059,   0.59159,   0.59259,   0.59359,   0.59459,   0.5956,   0.5966,
0.5976,   0.5986,   0.5996,
           0.6006,   0.6016,   0.6026,   0.6036,   0.6046,   0.60561,   0.60
661,   0.60761,   0.60861,   0.60961,   0.61061,   0.61161,   0.61261,   0.6136
1,   0.61461,   0.61562,   0.61662,   0.61762,   0.61862,   0.61962,   0.62062,
0.62162,   0.62262,   0.62362,
          0.62462,   0.62563,   0.62663,   0.62763,   0.62863,   0.62963,   0.63
063,   0.63163,   0.63263,   0.63363,   0.63463,   0.63564,   0.63664,   0.6376
4,   0.63864,   0.63964,   0.64064,   0.64164,   0.64264,   0.64364,   0.64464,
0.64565,   0.64665,   0.64765,
          0.64865,   0.64965,   0.65065,   0.65165,   0.65265,   0.65365,   0.65
465,   0.65566,   0.65666,   0.65766,   0.65866,   0.65966,   0.66066,   0.6616
6,   0.66266,   0.66366,   0.66466,   0.66567,   0.66667,   0.66767,   0.66867,
0.66967,   0.67067,   0.67167,
          0.67267,   0.67367,   0.67467,   0.67568,   0.67668,   0.67768,   0.67
868,   0.67968,   0.68068,   0.68168,   0.68268,   0.68368,   0.68468,   0.6856
9,   0.68669,   0.68769,   0.68869,   0.68969,   0.69069,   0.69169,   0.69269,
0.69369,   0.69469,   0.6957,
           0.6967,   0.6977,   0.6987,   0.6997,   0.7007,   0.7017,   0.7
027,   0.7037,   0.7047,   0.70571,   0.70671,   0.70771,   0.70871,   0.7097
1,   0.71071,   0.71171,   0.71271,   0.71371,   0.71471,   0.71572,   0.71672,
0.71772,   0.71872,   0.71972,
          0.72072,   0.72172,   0.72272,   0.72372,   0.72472,   0.72573,   0.72
673,   0.72773,   0.72873,   0.72973,   0.73073,   0.73173,   0.73273,   0.7337
3,   0.73473,   0.73574,   0.73674,   0.73774,   0.73874,   0.73974,   0.74074,
0.74174,   0.74274,   0.74374,

```
          0.74474,     0.74575,     0.74675,     0.74775,     0.74875,     0.74975,     0.75
075,     0.75175,     0.75275,     0.75375,     0.75475,     0.75576,     0.75676,     0.7577
6,     0.75876,     0.75976,     0.76076,     0.76176,     0.76276,     0.76376,     0.76476,
0.76577,     0.76677,     0.76777,
          0.76877,     0.76977,     0.77077,     0.77177,     0.77277,     0.77377,     0.77
477,     0.77578,     0.77678,     0.77778,     0.77878,     0.77978,     0.78078,     0.7817
8,     0.78278,     0.78378,     0.78478,     0.78579,     0.78679,     0.78779,     0.78879,
0.78979,     0.79079,     0.79179,
          0.79279,     0.79379,     0.79479,      0.7958,      0.7968,      0.7978,      0.7
988,      0.7998,      0.8008,      0.8018,      0.8028,      0.8038,      0.8048,     0.8058
1,     0.80681,     0.80781,     0.80881,     0.80981,     0.81081,     0.81181,     0.81281,
0.81381,     0.81481,     0.81582,
          0.81682,     0.81782,     0.81882,     0.81982,     0.82082,     0.82182,     0.82
282,     0.82382,     0.82482,     0.82583,     0.82683,     0.82783,     0.82883,     0.8298
3,     0.83083,     0.83183,     0.83283,     0.83383,     0.83483,     0.83584,     0.83684,
0.83784,     0.83884,     0.83984,
          0.84084,     0.84184,     0.84284,     0.84384,     0.84484,     0.84585,     0.84
685,     0.84785,     0.84885,     0.84985,     0.85085,     0.85185,     0.85285,     0.8538
5,     0.85485,     0.85586,     0.85686,     0.85786,     0.85886,     0.85986,     0.86086,
0.86186,     0.86286,     0.86386,
          0.86486,     0.86587,     0.86687,     0.86787,     0.86887,     0.86987,     0.87
087,     0.87187,     0.87287,     0.87387,     0.87487,     0.87588,     0.87688,     0.8778
8,     0.87888,     0.87988,     0.88088,     0.88188,     0.88288,     0.88388,     0.88488,
0.88589,     0.88689,     0.88789,
          0.88889,     0.88989,     0.89089,     0.89189,     0.89289,     0.89389,     0.89
489,      0.8959,      0.8969,      0.8979,      0.8989,      0.8999,      0.9009,      0.901
9,      0.9029,      0.9039,      0.9049,     0.90591,     0.90691,     0.90791,     0.90891,
0.90991,     0.91091,     0.91191,
          0.91291,     0.91391,     0.91491,     0.91592,     0.91692,     0.91792,     0.91
892,     0.91992,     0.92092,     0.92192,     0.92292,     0.92392,     0.92492,     0.9259
3,     0.92693,     0.92793,     0.92893,     0.92993,     0.93093,     0.93193,     0.93293,
0.93393,     0.93493,     0.93594,
          0.93694,     0.93794,     0.93894,     0.93994,     0.94094,     0.94194,     0.94
294,     0.94394,     0.94494,     0.94595,     0.94695,     0.94795,     0.94895,     0.9499
5,     0.95095,     0.95195,     0.95295,     0.95395,     0.95495,     0.95596,     0.95696,
0.95796,     0.95896,     0.95996,
          0.96096,     0.96196,     0.96296,     0.96396,     0.96496,     0.96597,     0.96
697,     0.96797,     0.96897,     0.96997,     0.97097,     0.97197,     0.97297,     0.9739
7,     0.97497,     0.97598,     0.97698,     0.97798,     0.97898,     0.97998,     0.98098,
0.98198,     0.98298,     0.98398,
          0.98498,     0.98599,     0.98699,     0.98799,     0.98899,     0.98999,     0.99
099,     0.99199,     0.99299,     0.99399,     0.99499,       0.996,       0.997,       0.99
8,       0.999,           1]), array([[          1,           1,           1, ...,  6.3221e-0
5,  3.1611e-05,           0],
       [        0.5,         0.5,         0.5, ...,  0.00010935,  5.4675e-05,           0],
       [    0.66667,     0.66667,     0.66667, ...,   0.0012833,  0.00064167,           0]]),
'Recall', 'Precision'], [array([          0,    0.001001,    0.002002,    0.003003,    0.0040
04,    0.005005,    0.006006,    0.007007,    0.008008,    0.009009,     0.01001,    0.01101
1,    0.012012,    0.013013,    0.014014,    0.015015,    0.016016,    0.017017,    0.018018,
0.019019,     0.02002,    0.021021,    0.022022,    0.023023,
          0.024024,    0.025025,    0.026026,    0.027027,    0.028028,    0.029029,     0.03
003,    0.031031,    0.032032,    0.033033,    0.034034,    0.035035,    0.036036,    0.03703
7,    0.038038,    0.039039,     0.04004,    0.041041,    0.042042,    0.043043,    0.044044,
0.045045,    0.046046,    0.047047,
          0.048048,    0.049049,     0.05005,    0.051051,    0.052052,    0.053053,    0.054
054,    0.055055,    0.056056,    0.057057,    0.058058,    0.059059,     0.06006,    0.06106
1,    0.062062,    0.063063,    0.064064,    0.065065,    0.066066,    0.067067,    0.068068,
0.069069,     0.07007,    0.071071,
          0.072072,    0.073073,    0.074074,    0.075075,    0.076076,    0.077077,    0.078
078,    0.079079,     0.08008,    0.081081,    0.082082,    0.083083,    0.084084,    0.08508
5,    0.086086,    0.087087,    0.088088,    0.089089,     0.09009,    0.091091,    0.092092,
0.093093,    0.094094,    0.095095,
          0.096096,    0.097097,    0.098098,    0.099099,      0.1001,      0.1011,      0.1
021,      0.1031,      0.1041,     0.10511,     0.10611,     0.10711,     0.10811,     0.1091
1,     0.11011,     0.11111,     0.11211,     0.11311,     0.11411,     0.11512,     0.11612,
0.11712,     0.11812,     0.11912,
```

0.12012, 0.12112, 0.12212, 0.12312, 0.12412, 0.12513, 0.12
613, 0.12713, 0.12813, 0.12913, 0.13013, 0.13113, 0.13213, 0.1331
3, 0.13413, 0.13514, 0.13614, 0.13714, 0.13814, 0.13914, 0.14014,
0.14114, 0.14214, 0.14314,
0.14414, 0.14515, 0.14615, 0.14715, 0.14815, 0.14915, 0.15
015, 0.15115, 0.15215, 0.15315, 0.15415, 0.15516, 0.15616, 0.1571
6, 0.15816, 0.15916, 0.16016, 0.16116, 0.16216, 0.16316, 0.16416,
0.16517, 0.16617, 0.16717,
0.16817, 0.16917, 0.17017, 0.17117, 0.17217, 0.17317, 0.17
417, 0.17518, 0.17618, 0.17718, 0.17818, 0.17918, 0.18018, 0.1811
8, 0.18218, 0.18318, 0.18418, 0.18519, 0.18619, 0.18719, 0.18819,
0.18919, 0.19019, 0.19119,
0.19219, 0.19319, 0.19419, 0.1952, 0.1962, 0.1972, 0.1
982, 0.1992, 0.2002, 0.2012, 0.2022, 0.2032, 0.2042, 0.2052
1, 0.20621, 0.20721, 0.20821, 0.20921, 0.21021, 0.21121, 0.21221,
0.21321, 0.21421, 0.21522,
0.21622, 0.21722, 0.21822, 0.21922, 0.22022, 0.22122, 0.22
222, 0.22322, 0.22422, 0.22523, 0.22623, 0.22723, 0.22823, 0.2292
3, 0.23023, 0.23123, 0.23223, 0.23323, 0.23423, 0.23524, 0.23624,
0.23724, 0.23824, 0.23924,
0.24024, 0.24124, 0.24224, 0.24324, 0.24424, 0.24525, 0.24
625, 0.24725, 0.24825, 0.24925, 0.25025, 0.25125, 0.25225, 0.2532
5, 0.25425, 0.25526, 0.25626, 0.25726, 0.25826, 0.25926, 0.26026,
0.26126, 0.26226, 0.26326,
0.26426, 0.26527, 0.26627, 0.26727, 0.26827, 0.26927, 0.27
027, 0.27127, 0.27227, 0.27327, 0.27427, 0.27528, 0.27628, 0.2772
8, 0.27828, 0.27928, 0.28028, 0.28128, 0.28228, 0.28328, 0.28428,
0.28529, 0.28629, 0.28729,
0.28829, 0.28929, 0.29029, 0.29129, 0.29229, 0.29329, 0.29
429, 0.2953, 0.2963, 0.2973, 0.2983, 0.2993, 0.3003, 0.301
3, 0.3023, 0.3033, 0.3043, 0.30531, 0.30631, 0.30731, 0.30831,
0.30931, 0.31031, 0.31131,
0.31231, 0.31331, 0.31431, 0.31532, 0.31632, 0.31732, 0.31
832, 0.31932, 0.32032, 0.32132, 0.32232, 0.32332, 0.32432, 0.3253
3, 0.32633, 0.32733, 0.32833, 0.32933, 0.33033, 0.33133, 0.33233,
0.33333, 0.33433, 0.33534,
0.33634, 0.33734, 0.33834, 0.33934, 0.34034, 0.34134, 0.34
234, 0.34334, 0.34434, 0.34535, 0.34635, 0.34735, 0.34835, 0.3493
5, 0.35035, 0.35135, 0.35235, 0.35335, 0.35435, 0.35536, 0.35636,
0.35736, 0.35836, 0.35936,
0.36036, 0.36136, 0.36236, 0.36336, 0.36436, 0.36537, 0.36
637, 0.36737, 0.36837, 0.36937, 0.37037, 0.37137, 0.37237, 0.3733
7, 0.37437, 0.37538, 0.37638, 0.37738, 0.37838, 0.37938, 0.38038,
0.38138, 0.38238, 0.38338,
0.38438, 0.38539, 0.38639, 0.38739, 0.38839, 0.38939, 0.39
039, 0.39139, 0.39239, 0.39339, 0.39439, 0.3954, 0.3964, 0.397
4, 0.3984, 0.3994, 0.4004, 0.4014, 0.4024, 0.4034, 0.4044,
0.40541, 0.40641, 0.40741,
0.40841, 0.40941, 0.41041, 0.41141, 0.41241, 0.41341, 0.41
441, 0.41542, 0.41642, 0.41742, 0.41842, 0.41942, 0.42042, 0.4214
2, 0.42242, 0.42342, 0.42442, 0.42543, 0.42643, 0.42743, 0.42843,
0.42943, 0.43043, 0.43143,
0.43243, 0.43343, 0.43443, 0.43544, 0.43644, 0.43744, 0.43
844, 0.43944, 0.44044, 0.44144, 0.44244, 0.44344, 0.44444, 0.4454
5, 0.44645, 0.44745, 0.44845, 0.44945, 0.45045, 0.45145, 0.45245,
0.45345, 0.45445, 0.45546,
0.45646, 0.45746, 0.45846, 0.45946, 0.46046, 0.46146, 0.46
246, 0.46346, 0.46446, 0.46547, 0.46647, 0.46747, 0.46847, 0.4694
7, 0.47047, 0.47147, 0.47247, 0.47347, 0.47447, 0.47548, 0.47648,
0.47748, 0.47848, 0.47948,
0.48048, 0.48148, 0.48248, 0.48348, 0.48448, 0.48549, 0.48
649, 0.48749, 0.48849, 0.48949, 0.49049, 0.49149, 0.49249, 0.4934
9, 0.49449, 0.4955, 0.4965, 0.4975, 0.4985, 0.4995, 0.5005,
0.5015, 0.5025, 0.5035,
0.5045, 0.50551, 0.50651, 0.50751, 0.50851, 0.50951, 0.51
051, 0.51151, 0.51251, 0.51351, 0.51451, 0.51552, 0.51652, 0.5175

2,   0.51852,   0.51952,   0.52052,   0.52152,   0.52252,   0.52352,   0.52452,
0.52553,   0.52653,   0.52753,
          0.52853,   0.52953,   0.53053,   0.53153,   0.53253,   0.53353,   0.53
453,   0.53554,   0.53654,   0.53754,   0.53854,   0.53954,   0.54054,   0.5415
4,   0.54254,   0.54354,   0.54454,   0.54555,   0.54655,   0.54755,   0.54855,
0.54955,   0.55055,   0.55155,
          0.55255,   0.55355,   0.55455,   0.55556,   0.55656,   0.55756,   0.55
856,   0.55956,   0.56056,   0.56156,   0.56256,   0.56356,   0.56456,   0.5655
7,   0.56657,   0.56757,   0.56857,   0.56957,   0.57057,   0.57157,   0.57257,
0.57357,   0.57457,   0.57558,
          0.57658,   0.57758,   0.57858,   0.57958,   0.58058,   0.58158,   0.58
258,   0.58358,   0.58458,   0.58559,   0.58659,   0.58759,   0.58859,   0.5895
9,   0.59059,   0.59159,   0.59259,   0.59359,   0.59459,   0.5956,   0.5966,
0.5976,   0.5986,   0.5996,
           0.6006,   0.6016,   0.6026,   0.6036,   0.6046,   0.60561,   0.60
661,   0.60761,   0.60861,   0.60961,   0.61061,   0.61161,   0.61261,   0.6136
1,   0.61461,   0.61562,   0.61662,   0.61762,   0.61862,   0.61962,   0.62062,
0.62162,   0.62262,   0.62362,
          0.62462,   0.62563,   0.62663,   0.62763,   0.62863,   0.62963,   0.63
063,   0.63163,   0.63263,   0.63363,   0.63463,   0.63564,   0.63664,   0.6376
4,   0.63864,   0.63964,   0.64064,   0.64164,   0.64264,   0.64364,   0.64464,
0.64565,   0.64665,   0.64765,
          0.64865,   0.64965,   0.65065,   0.65165,   0.65265,   0.65365,   0.65
465,   0.65566,   0.65666,   0.65766,   0.65866,   0.65966,   0.66066,   0.6616
6,   0.66266,   0.66366,   0.66466,   0.66567,   0.66667,   0.66767,   0.66867,
0.66967,   0.67067,   0.67167,
          0.67267,   0.67367,   0.67467,   0.67568,   0.67668,   0.67768,   0.67
868,   0.67968,   0.68068,   0.68168,   0.68268,   0.68368,   0.68468,   0.6856
9,   0.68669,   0.68769,   0.68869,   0.68969,   0.69069,   0.69169,   0.69269,
0.69369,   0.69469,   0.6957,
           0.6967,   0.6977,   0.6987,   0.6997,   0.7007,   0.7017,   0.7
027,   0.7037,   0.7047,   0.70571,   0.70671,   0.70771,   0.70871,   0.7097
1,   0.71071,   0.71171,   0.71271,   0.71371,   0.71471,   0.71572,   0.71672,
0.71772,   0.71872,   0.71972,
          0.72072,   0.72172,   0.72272,   0.72372,   0.72472,   0.72573,   0.72
673,   0.72773,   0.72873,   0.72973,   0.73073,   0.73173,   0.73273,   0.7337
3,   0.73473,   0.73574,   0.73674,   0.73774,   0.73874,   0.73974,   0.74074,
0.74174,   0.74274,   0.74374,
          0.74474,   0.74575,   0.74675,   0.74775,   0.74875,   0.74975,   0.75
075,   0.75175,   0.75275,   0.75375,   0.75475,   0.75576,   0.75676,   0.7577
6,   0.75876,   0.75976,   0.76076,   0.76176,   0.76276,   0.76376,   0.76476,
0.76577,   0.76677,   0.76777,
          0.76877,   0.76977,   0.77077,   0.77177,   0.77277,   0.77377,   0.77
477,   0.77578,   0.77678,   0.77778,   0.77878,   0.77978,   0.78078,   0.7817
8,   0.78278,   0.78378,   0.78478,   0.78579,   0.78679,   0.78779,   0.78879,
0.78979,   0.79079,   0.79179,
          0.79279,   0.79379,   0.79479,   0.7958,   0.7968,   0.7978,   0.7
988,   0.7998,   0.8008,   0.8018,   0.8028,   0.8038,   0.8048,   0.8058
1,   0.80681,   0.80781,   0.80881,   0.80981,   0.81081,   0.81181,   0.81281,
0.81381,   0.81481,   0.81582,
          0.81682,   0.81782,   0.81882,   0.81982,   0.82082,   0.82182,   0.82
282,   0.82382,   0.82482,   0.82583,   0.82683,   0.82783,   0.82883,   0.8298
3,   0.83083,   0.83183,   0.83283,   0.83383,   0.83483,   0.83584,   0.83684,
0.83784,   0.83884,   0.83984,
          0.84084,   0.84184,   0.84284,   0.84384,   0.84484,   0.84585,   0.84
685,   0.84785,   0.84885,   0.84985,   0.85085,   0.85185,   0.85285,   0.8538
5,   0.85485,   0.85586,   0.85686,   0.85786,   0.85886,   0.85986,   0.86086,
0.86186,   0.86286,   0.86386,
          0.86486,   0.86587,   0.86687,   0.86787,   0.86887,   0.86987,   0.87
087,   0.87187,   0.87287,   0.87387,   0.87487,   0.87588,   0.87688,   0.8778
8,   0.87888,   0.87988,   0.88088,   0.88188,   0.88288,   0.88388,   0.88488,
0.88589,   0.88689,   0.88789,
          0.88889,   0.88989,   0.89089,   0.89189,   0.89289,   0.89389,   0.89
489,   0.8959,   0.8969,   0.8979,   0.8989,   0.8999,   0.9009,   0.901
9,   0.9029,   0.9039,   0.9049,   0.90591,   0.90691,   0.90791,   0.90891,
0.90991,   0.91091,   0.91191,

```
          0.91291,     0.91391,     0.91491,     0.91592,     0.91692,     0.91792,     0.91
892,     0.91992,     0.92092,     0.92192,     0.92292,     0.92392,     0.92492,     0.9259
3,     0.92693,     0.92793,     0.92893,     0.92993,     0.93093,     0.93193,     0.93293,
0.93393,     0.93493,     0.93594,
          0.93694,     0.93794,     0.93894,     0.93994,     0.94094,     0.94194,     0.94
294,     0.94394,     0.94494,     0.94595,     0.94695,     0.94795,     0.94895,     0.9499
5,     0.95095,     0.95195,     0.95295,     0.95395,     0.95495,     0.95596,     0.95696,
0.95796,     0.95896,     0.95996,
          0.96096,     0.96196,     0.96296,     0.96396,     0.96496,     0.96597,     0.96
697,     0.96797,     0.96897,     0.96997,     0.97097,     0.97197,     0.97297,     0.9739
7,     0.97497,     0.97598,     0.97698,     0.97798,     0.97898,     0.97998,     0.98098,
0.98198,     0.98298,     0.98398,
          0.98498,     0.98599,     0.98699,     0.98799,     0.98899,     0.98999,     0.99
099,     0.99199,     0.99299,     0.99399,     0.99499,       0.996,       0.997,       0.99
8,       0.999,           1]), array([[  0.0089753,   0.0089753,     0.01076, ...,
0,           0,           0],
       [   0.020544,    0.020544,    0.023079, ...,           0,           0,           0],
       [   0.049948,    0.049948,    0.057153, ...,           0,           0,           0]]),
'Confidence', 'F1'], [array([          0,    0.001001,    0.002002,    0.003003,    0.004004,
0.005005,    0.006006,    0.007007,    0.008008,    0.009009,     0.01001,    0.011011,    0.
012012,    0.013013,    0.014014,    0.015015,    0.016016,    0.017017,    0.018018,    0.01
9019,     0.02002,    0.021021,    0.022022,    0.023023,
          0.024024,    0.025025,    0.026026,    0.027027,    0.028028,    0.029029,     0.03
003,    0.031031,    0.032032,    0.033033,    0.034034,    0.035035,    0.036036,    0.03703
7,    0.038038,    0.039039,     0.04004,    0.041041,    0.042042,    0.043043,    0.044044,
0.045045,    0.046046,    0.047047,
          0.048048,    0.049049,     0.05005,    0.051051,    0.052052,    0.053053,    0.054
054,    0.055055,    0.056056,    0.057057,    0.058058,    0.059059,     0.06006,    0.06106
1,    0.062062,    0.063063,    0.064064,    0.065065,    0.066066,    0.067067,    0.068068,
0.069069,     0.07007,    0.071071,
          0.072072,    0.073073,    0.074074,    0.075075,    0.076076,    0.077077,    0.078
078,    0.079079,     0.08008,    0.081081,    0.082082,    0.083083,    0.084084,    0.08508
5,    0.086086,    0.087087,    0.088088,    0.089089,     0.09009,    0.091091,    0.092092,
0.093093,    0.094094,    0.095095,
          0.096096,    0.097097,    0.098098,    0.099099,      0.1001,      0.1011,      0.1
021,      0.1031,      0.1041,     0.10511,     0.10611,     0.10711,     0.10811,     0.1091
1,     0.11011,     0.11111,     0.11211,     0.11311,     0.11411,     0.11512,     0.11612,
0.11712,     0.11812,     0.11912,
          0.12012,     0.12112,     0.12212,     0.12312,     0.12412,     0.12513,     0.12
613,     0.12713,     0.12813,     0.12913,     0.13013,     0.13113,     0.13213,     0.1331
3,     0.13413,     0.13514,     0.13614,     0.13714,     0.13814,     0.13914,     0.14014,
0.14114,     0.14214,     0.14314,
          0.14414,     0.14515,     0.14615,     0.14715,     0.14815,     0.14915,     0.15
015,     0.15115,     0.15215,     0.15315,     0.15415,     0.15516,     0.15616,     0.1571
6,     0.15816,     0.15916,     0.16016,     0.16116,     0.16216,     0.16316,     0.16416,
0.16517,     0.16617,     0.16717,
          0.16817,     0.16917,     0.17017,     0.17117,     0.17217,     0.17317,     0.17
417,     0.17518,     0.17618,     0.17718,     0.17818,     0.17918,     0.18018,     0.1811
8,     0.18218,     0.18318,     0.18418,     0.18519,     0.18619,     0.18719,     0.18819,
0.18919,     0.19019,     0.19119,
          0.19219,     0.19319,     0.19419,      0.1952,      0.1962,      0.1972,      0.1
982,      0.1992,      0.2002,      0.2012,      0.2022,      0.2032,      0.2042,     0.2052
1,     0.20621,     0.20721,     0.20821,     0.20921,     0.21021,     0.21121,     0.21221,
0.21321,     0.21421,     0.21522,
          0.21622,     0.21722,     0.21822,     0.21922,     0.22022,     0.22122,     0.22
222,     0.22322,     0.22422,     0.22523,     0.22623,     0.22723,     0.22823,     0.2292
3,     0.23023,     0.23123,     0.23223,     0.23323,     0.23423,     0.23524,     0.23624,
0.23724,     0.23824,     0.23924,
          0.24024,     0.24124,     0.24224,     0.24324,     0.24424,     0.24525,     0.24
625,     0.24725,     0.24825,     0.24925,     0.25025,     0.25125,     0.25225,     0.2532
5,     0.25425,     0.25526,     0.25626,     0.25726,     0.25826,     0.25926,     0.26026,
0.26126,     0.26226,     0.26326,
          0.26426,     0.26527,     0.26627,     0.26727,     0.26827,     0.26927,     0.27
027,     0.27127,     0.27227,     0.27327,     0.27427,     0.27528,     0.27628,     0.2772
8,     0.27828,     0.27928,     0.28028,     0.28128,     0.28228,     0.28328,     0.28428,
0.28529,     0.28629,     0.28729,
```

0.28829, 0.28929, 0.29029, 0.29129, 0.29229, 0.29329, 0.29
429, 0.2953, 0.2963, 0.2973, 0.2983, 0.2993, 0.3003, 0.301
3, 0.3023, 0.3033, 0.3043, 0.30531, 0.30631, 0.30731, 0.30831,
0.30931, 0.31031, 0.31131,
0.31231, 0.31331, 0.31431, 0.31532, 0.31632, 0.31732, 0.31
832, 0.31932, 0.32032, 0.32132, 0.32232, 0.32332, 0.32432, 0.3253
3, 0.32633, 0.32733, 0.32833, 0.32933, 0.33033, 0.33133, 0.33233,
0.33333, 0.33433, 0.33534,
0.33634, 0.33734, 0.33834, 0.33934, 0.34034, 0.34134, 0.34
234, 0.34334, 0.34434, 0.34535, 0.34635, 0.34735, 0.34835, 0.3493
5, 0.35035, 0.35135, 0.35235, 0.35335, 0.35435, 0.35536, 0.35636,
0.35736, 0.35836, 0.35936,
0.36036, 0.36136, 0.36236, 0.36336, 0.36436, 0.36537, 0.36
637, 0.36737, 0.36837, 0.36937, 0.37037, 0.37137, 0.37237, 0.3733
7, 0.37437, 0.37538, 0.37638, 0.37738, 0.37838, 0.37938, 0.38038,
0.38138, 0.38238, 0.38338,
0.38438, 0.38539, 0.38639, 0.38739, 0.38839, 0.38939, 0.39
039, 0.39139, 0.39239, 0.39339, 0.39439, 0.3954, 0.3964, 0.397
4, 0.3984, 0.3994, 0.4004, 0.4014, 0.4024, 0.4034, 0.4044,
0.40541, 0.40641, 0.40741,
0.40841, 0.40941, 0.41041, 0.41141, 0.41241, 0.41341, 0.41
441, 0.41542, 0.41642, 0.41742, 0.41842, 0.41942, 0.42042, 0.4214
2, 0.42242, 0.42342, 0.42442, 0.42543, 0.42643, 0.42743, 0.42843,
0.42943, 0.43043, 0.43143,
0.43243, 0.43343, 0.43443, 0.43544, 0.43644, 0.43744, 0.43
844, 0.43944, 0.44044, 0.44144, 0.44244, 0.44344, 0.44444, 0.4454
5, 0.44645, 0.44745, 0.44845, 0.44945, 0.45045, 0.45145, 0.45245,
0.45345, 0.45445, 0.45546,
0.45646, 0.45746, 0.45846, 0.45946, 0.46046, 0.46146, 0.46
246, 0.46346, 0.46446, 0.46547, 0.46647, 0.46747, 0.46847, 0.4694
7, 0.47047, 0.47147, 0.47247, 0.47347, 0.47447, 0.47548, 0.47648,
0.47748, 0.47848, 0.47948,
0.48048, 0.48148, 0.48248, 0.48348, 0.48448, 0.48549, 0.48
649, 0.48749, 0.48849, 0.48949, 0.49049, 0.49149, 0.49249, 0.4934
9, 0.49449, 0.4955, 0.4965, 0.4975, 0.4985, 0.4995, 0.5005,
0.5015, 0.5025, 0.5035,
0.5045, 0.50551, 0.50651, 0.50751, 0.50851, 0.50951, 0.51
051, 0.51151, 0.51251, 0.51351, 0.51451, 0.51552, 0.51652, 0.5175
2, 0.51852, 0.51952, 0.52052, 0.52152, 0.52252, 0.52352, 0.52452,
0.52553, 0.52653, 0.52753,
0.52853, 0.52953, 0.53053, 0.53153, 0.53253, 0.53353, 0.53
453, 0.53554, 0.53654, 0.53754, 0.53854, 0.53954, 0.54054, 0.5415
4, 0.54254, 0.54354, 0.54454, 0.54555, 0.54655, 0.54755, 0.54855,
0.54955, 0.55055, 0.55155,
0.55255, 0.55355, 0.55455, 0.55556, 0.55656, 0.55756, 0.55
856, 0.55956, 0.56056, 0.56156, 0.56256, 0.56356, 0.56456, 0.5655
7, 0.56657, 0.56757, 0.56857, 0.56957, 0.57057, 0.57157, 0.57257,
0.57357, 0.57457, 0.57558,
0.57658, 0.57758, 0.57858, 0.57958, 0.58058, 0.58158, 0.58
258, 0.58358, 0.58458, 0.58559, 0.58659, 0.58759, 0.58859, 0.5895
9, 0.59059, 0.59159, 0.59259, 0.59359, 0.59459, 0.5956, 0.5966,
0.5976, 0.5986, 0.5996,
0.6006, 0.6016, 0.6026, 0.6036, 0.6046, 0.60561, 0.60
661, 0.60761, 0.60861, 0.60961, 0.61061, 0.61161, 0.61261, 0.6136
1, 0.61461, 0.61562, 0.61662, 0.61762, 0.61862, 0.61962, 0.62062,
0.62162, 0.62262, 0.62362,
0.62462, 0.62563, 0.62663, 0.62763, 0.62863, 0.62963, 0.63
063, 0.63163, 0.63263, 0.63363, 0.63463, 0.63564, 0.63664, 0.6376
4, 0.63864, 0.63964, 0.64064, 0.64164, 0.64264, 0.64364, 0.64464,
0.64565, 0.64665, 0.64765,
0.64865, 0.64965, 0.65065, 0.65165, 0.65265, 0.65365, 0.65
465, 0.65566, 0.65666, 0.65766, 0.65866, 0.65966, 0.66066, 0.6616
6, 0.66266, 0.66366, 0.66466, 0.66567, 0.66667, 0.66767, 0.66867,
0.66967, 0.67067, 0.67167,
0.67267, 0.67367, 0.67467, 0.67568, 0.67668, 0.67768, 0.67
868, 0.67968, 0.68068, 0.68168, 0.68268, 0.68368, 0.68468, 0.6856

```
9,     0.68669,     0.68769,     0.68869,     0.68969,     0.69069,     0.69169,     0.69269,
0.69369,     0.69469,      0.6957,
          0.6967,      0.6977,      0.6987,      0.6997,      0.7007,      0.7017,      0.7
027,      0.7037,      0.7047,     0.70571,     0.70671,     0.70771,     0.70871,     0.7097
1,     0.71071,     0.71171,     0.71271,     0.71371,     0.71471,     0.71572,     0.71672,
0.71772,     0.71872,     0.71972,
          0.72072,     0.72172,     0.72272,     0.72372,     0.72472,     0.72573,     0.72
673,     0.72773,     0.72873,     0.72973,     0.73073,     0.73173,     0.73273,     0.7337
3,     0.73473,     0.73574,     0.73674,     0.73774,     0.73874,     0.73974,     0.74074,
0.74174,     0.74274,     0.74374,
          0.74474,     0.74575,     0.74675,     0.74775,     0.74875,     0.74975,     0.75
075,     0.75175,     0.75275,     0.75375,     0.75475,     0.75576,     0.75676,     0.7577
6,     0.75876,     0.75976,     0.76076,     0.76176,     0.76276,     0.76376,     0.76476,
0.76577,     0.76677,     0.76777,
          0.76877,     0.76977,     0.77077,     0.77177,     0.77277,     0.77377,     0.77
477,     0.77578,     0.77678,     0.77778,     0.77878,     0.77978,     0.78078,     0.7817
8,     0.78278,     0.78378,     0.78478,     0.78579,     0.78679,     0.78779,     0.78879,
0.78979,     0.79079,     0.79179,
          0.79279,     0.79379,     0.79479,      0.7958,      0.7968,      0.7978,      0.7
988,      0.7998,      0.8008,      0.8018,      0.8028,      0.8038,      0.8048,     0.8058
1,     0.80681,     0.80781,     0.80881,     0.80981,     0.81081,     0.81181,     0.81281,
0.81381,     0.81481,     0.81582,
          0.81682,     0.81782,     0.81882,     0.81982,     0.82082,     0.82182,     0.82
282,     0.82382,     0.82482,     0.82583,     0.82683,     0.82783,     0.82883,     0.8298
3,     0.83083,     0.83183,     0.83283,     0.83383,     0.83483,     0.83584,     0.83684,
0.83784,     0.83884,     0.83984,
          0.84084,     0.84184,     0.84284,     0.84384,     0.84484,     0.84585,     0.84
685,     0.84785,     0.84885,     0.84985,     0.85085,     0.85185,     0.85285,     0.8538
5,     0.85485,     0.85586,     0.85686,     0.85786,     0.85886,     0.85986,     0.86086,
0.86186,     0.86286,     0.86386,
          0.86486,     0.86587,     0.86687,     0.86787,     0.86887,     0.86987,     0.87
087,     0.87187,     0.87287,     0.87387,     0.87487,     0.87588,     0.87688,     0.8778
8,     0.87888,     0.87988,     0.88088,     0.88188,     0.88288,     0.88388,     0.88488,
0.88589,     0.88689,     0.88789,
          0.88889,     0.88989,     0.89089,     0.89189,     0.89289,     0.89389,     0.89
489,      0.8959,      0.8969,      0.8979,      0.8989,      0.8999,      0.9009,      0.901
9,      0.9029,      0.9039,      0.9049,     0.90591,     0.90691,     0.90791,     0.90891,
0.90991,     0.91091,     0.91191,
          0.91291,     0.91391,     0.91491,     0.91592,     0.91692,     0.91792,     0.91
892,     0.91992,     0.92092,     0.92192,     0.92292,     0.92392,     0.92492,     0.9259
3,     0.92693,     0.92793,     0.92893,     0.92993,     0.93093,     0.93193,     0.93293,
0.93393,     0.93493,     0.93594,
          0.93694,     0.93794,     0.93894,     0.93994,     0.94094,     0.94194,     0.94
294,     0.94394,     0.94494,     0.94595,     0.94695,     0.94795,     0.94895,     0.9499
5,     0.95095,     0.95195,     0.95295,     0.95395,     0.95495,     0.95596,     0.95696,
0.95796,     0.95896,     0.95996,
          0.96096,     0.96196,     0.96296,     0.96396,     0.96496,     0.96597,     0.96
697,     0.96797,     0.96897,     0.96997,     0.97097,     0.97197,     0.97297,     0.9739
7,     0.97497,     0.97598,     0.97698,     0.97798,     0.97898,     0.97998,     0.98098,
0.98198,     0.98298,     0.98398,
          0.98498,     0.98599,     0.98699,     0.98799,     0.98899,     0.98999,     0.99
099,     0.99199,     0.99299,     0.99399,     0.99499,       0.996,       0.997,       0.99
8,       0.999,           1]), array([[  0.0045113,   0.0045113,   0.0054141, ...,
1,           1,           1],
       [   0.010404,    0.010404,    0.011706, ...,           1,           1,           1],
       [   0.025641,    0.025641,    0.029453, ...,           1,           1,           1]]),
'Confidence', 'Precision'], [array([          0,    0.001001,    0.002002,    0.003003,    0.
004004,    0.005005,    0.006006,    0.007007,    0.008008,    0.009009,     0.01001,    0.01
1011,    0.012012,    0.013013,    0.014014,    0.015015,    0.016016,    0.017017,    0.0180
18,    0.019019,     0.02002,    0.021021,    0.022022,    0.023023,
          0.024024,    0.025025,    0.026026,    0.027027,    0.028028,    0.029029,     0.03
003,    0.031031,    0.032032,    0.033033,    0.034034,    0.035035,    0.036036,    0.03703
7,    0.038038,    0.039039,     0.04004,    0.041041,    0.042042,    0.043043,    0.044044,
0.045045,    0.046046,    0.047047,
          0.048048,    0.049049,     0.05005,    0.051051,    0.052052,    0.053053,    0.054
054,    0.055055,    0.056056,    0.057057,    0.058058,    0.059059,     0.06006,    0.06106
```

1, 0.062062, 0.063063, 0.064064, 0.065065, 0.066066, 0.067067, 0.068068,
0.069069, 0.07007, 0.071071,
0.072072, 0.073073, 0.074074, 0.075075, 0.076076, 0.077077, 0.078
078, 0.079079, 0.08008, 0.081081, 0.082082, 0.083083, 0.084084, 0.08508
5, 0.086086, 0.087087, 0.088088, 0.089089, 0.09009, 0.091091, 0.092092,
0.093093, 0.094094, 0.095095,
0.096096, 0.097097, 0.098098, 0.099099, 0.1001, 0.1011, 0.1
021, 0.1031, 0.1041, 0.10511, 0.10611, 0.10711, 0.10811, 0.1091
1, 0.11011, 0.11111, 0.11211, 0.11311, 0.11411, 0.11512, 0.11612,
0.11712, 0.11812, 0.11912,
0.12012, 0.12112, 0.12212, 0.12312, 0.12412, 0.12513, 0.12
613, 0.12713, 0.12813, 0.12913, 0.13013, 0.13113, 0.13213, 0.1331
3, 0.13413, 0.13514, 0.13614, 0.13714, 0.13814, 0.13914, 0.14014,
0.14114, 0.14214, 0.14314,
0.14414, 0.14515, 0.14615, 0.14715, 0.14815, 0.14915, 0.15
015, 0.15115, 0.15215, 0.15315, 0.15415, 0.15516, 0.15616, 0.1571
6, 0.15816, 0.15916, 0.16016, 0.16116, 0.16216, 0.16316, 0.16416,
0.16517, 0.16617, 0.16717,
0.16817, 0.16917, 0.17017, 0.17117, 0.17217, 0.17317, 0.17
417, 0.17518, 0.17618, 0.17718, 0.17818, 0.17918, 0.18018, 0.1811
8, 0.18218, 0.18318, 0.18418, 0.18519, 0.18619, 0.18719, 0.18819,
0.18919, 0.19019, 0.19119,
0.19219, 0.19319, 0.19419, 0.1952, 0.1962, 0.1972, 0.1
982, 0.1992, 0.2002, 0.2012, 0.2022, 0.2032, 0.2042, 0.2052
1, 0.20621, 0.20721, 0.20821, 0.20921, 0.21021, 0.21121, 0.21221,
0.21321, 0.21421, 0.21522,
0.21622, 0.21722, 0.21822, 0.21922, 0.22022, 0.22122, 0.22
222, 0.22322, 0.22422, 0.22523, 0.22623, 0.22723, 0.22823, 0.2292
3, 0.23023, 0.23123, 0.23223, 0.23323, 0.23423, 0.23524, 0.23624,
0.23724, 0.23824, 0.23924,
0.24024, 0.24124, 0.24224, 0.24324, 0.24424, 0.24525, 0.24
625, 0.24725, 0.24825, 0.24925, 0.25025, 0.25125, 0.25225, 0.2532
5, 0.25425, 0.25526, 0.25626, 0.25726, 0.25826, 0.25926, 0.26026,
0.26126, 0.26226, 0.26326,
0.26426, 0.26527, 0.26627, 0.26727, 0.26827, 0.26927, 0.27
027, 0.27127, 0.27227, 0.27327, 0.27427, 0.27528, 0.27628, 0.2772
8, 0.27828, 0.27928, 0.28028, 0.28128, 0.28228, 0.28328, 0.28428,
0.28529, 0.28629, 0.28729,
0.28829, 0.28929, 0.29029, 0.29129, 0.29229, 0.29329, 0.29
429, 0.2953, 0.2963, 0.2973, 0.2983, 0.2993, 0.3003, 0.301
3, 0.3023, 0.3033, 0.3043, 0.30531, 0.30631, 0.30731, 0.30831,
0.30931, 0.31031, 0.31131,
0.31231, 0.31331, 0.31431, 0.31532, 0.31632, 0.31732, 0.31
832, 0.31932, 0.32032, 0.32132, 0.32232, 0.32332, 0.32432, 0.3253
3, 0.32633, 0.32733, 0.32833, 0.32933, 0.33033, 0.33133, 0.33233,
0.33333, 0.33433, 0.33534,
0.33634, 0.33734, 0.33834, 0.33934, 0.34034, 0.34134, 0.34
234, 0.34334, 0.34434, 0.34535, 0.34635, 0.34735, 0.34835, 0.3493
5, 0.35035, 0.35135, 0.35235, 0.35335, 0.35435, 0.35536, 0.35636,
0.35736, 0.35836, 0.35936,
0.36036, 0.36136, 0.36236, 0.36336, 0.36436, 0.36537, 0.36
637, 0.36737, 0.36837, 0.36937, 0.37037, 0.37137, 0.37237, 0.3733
7, 0.37437, 0.37538, 0.37638, 0.37738, 0.37838, 0.37938, 0.38038,
0.38138, 0.38238, 0.38338,
0.38438, 0.38539, 0.38639, 0.38739, 0.38839, 0.38939, 0.39
039, 0.39139, 0.39239, 0.39339, 0.39439, 0.3954, 0.3964, 0.397
4, 0.3984, 0.3994, 0.4004, 0.4014, 0.4024, 0.4034, 0.4044,
0.40541, 0.40641, 0.40741,
0.40841, 0.40941, 0.41041, 0.41141, 0.41241, 0.41341, 0.41
441, 0.41542, 0.41642, 0.41742, 0.41842, 0.41942, 0.42042, 0.4214
2, 0.42242, 0.42342, 0.42442, 0.42543, 0.42643, 0.42743, 0.42843,
0.42943, 0.43043, 0.43143,
0.43243, 0.43343, 0.43443, 0.43544, 0.43644, 0.43744, 0.43
844, 0.43944, 0.44044, 0.44144, 0.44244, 0.44344, 0.44444, 0.4454
5, 0.44645, 0.44745, 0.44845, 0.44945, 0.45045, 0.45145, 0.45245,
0.45345, 0.45445, 0.45546,

0.45646, 0.45746, 0.45846, 0.45946, 0.46046, 0.46146, 0.46
246, 0.46346, 0.46446, 0.46547, 0.46647, 0.46747, 0.46847, 0.4694
7, 0.47047, 0.47147, 0.47247, 0.47347, 0.47447, 0.47548, 0.47648,
0.47748, 0.47848, 0.47948,
0.48048, 0.48148, 0.48248, 0.48348, 0.48448, 0.48549, 0.48
649, 0.48749, 0.48849, 0.48949, 0.49049, 0.49149, 0.49249, 0.4934
9, 0.49449, 0.4955, 0.4965, 0.4975, 0.4985, 0.4995, 0.5005,
0.5015, 0.5025, 0.5035,
0.5045, 0.50551, 0.50651, 0.50751, 0.50851, 0.50951, 0.51
051, 0.51151, 0.51251, 0.51351, 0.51451, 0.51552, 0.51652, 0.5175
2, 0.51852, 0.51952, 0.52052, 0.52152, 0.52252, 0.52352, 0.52452,
0.52553, 0.52653, 0.52753,
0.52853, 0.52953, 0.53053, 0.53153, 0.53253, 0.53353, 0.53
453, 0.53554, 0.53654, 0.53754, 0.53854, 0.53954, 0.54054, 0.5415
4, 0.54254, 0.54354, 0.54454, 0.54555, 0.54655, 0.54755, 0.54855,
0.54955, 0.55055, 0.55155,
0.55255, 0.55355, 0.55455, 0.55556, 0.55656, 0.55756, 0.55
856, 0.55956, 0.56056, 0.56156, 0.56256, 0.56356, 0.56456, 0.5655
7, 0.56657, 0.56757, 0.56857, 0.56957, 0.57057, 0.57157, 0.57257,
0.57357, 0.57457, 0.57558,
0.57658, 0.57758, 0.57858, 0.57958, 0.58058, 0.58158, 0.58
258, 0.58358, 0.58458, 0.58559, 0.58659, 0.58759, 0.58859, 0.5895
9, 0.59059, 0.59159, 0.59259, 0.59359, 0.59459, 0.5956, 0.5966,
0.5976, 0.5986, 0.5996,
0.6006, 0.6016, 0.6026, 0.6036, 0.6046, 0.60561, 0.60
661, 0.60761, 0.60861, 0.60961, 0.61061, 0.61161, 0.61261, 0.6136
1, 0.61461, 0.61562, 0.61662, 0.61762, 0.61862, 0.61962, 0.62062,
0.62162, 0.62262, 0.62362,
0.62462, 0.62563, 0.62663, 0.62763, 0.62863, 0.62963, 0.63
063, 0.63163, 0.63263, 0.63363, 0.63463, 0.63564, 0.63664, 0.6376
4, 0.63864, 0.63964, 0.64064, 0.64164, 0.64264, 0.64364, 0.64464,
0.64565, 0.64665, 0.64765,
0.64865, 0.64965, 0.65065, 0.65165, 0.65265, 0.65365, 0.65
465, 0.65566, 0.65666, 0.65766, 0.65866, 0.65966, 0.66066, 0.6616
6, 0.66266, 0.66366, 0.66466, 0.66567, 0.66667, 0.66767, 0.66867,
0.66967, 0.67067, 0.67167,
0.67267, 0.67367, 0.67467, 0.67568, 0.67668, 0.67768, 0.67
868, 0.67968, 0.68068, 0.68168, 0.68268, 0.68368, 0.68468, 0.6856
9, 0.68669, 0.68769, 0.68869, 0.68969, 0.69069, 0.69169, 0.69269,
0.69369, 0.69469, 0.6957,
0.6967, 0.6977, 0.6987, 0.6997, 0.7007, 0.7017, 0.7
027, 0.7037, 0.7047, 0.70571, 0.70671, 0.70771, 0.70871, 0.7097
1, 0.71071, 0.71171, 0.71271, 0.71371, 0.71471, 0.71572, 0.71672,
0.71772, 0.71872, 0.71972,
0.72072, 0.72172, 0.72272, 0.72372, 0.72472, 0.72573, 0.72
673, 0.72773, 0.72873, 0.72973, 0.73073, 0.73173, 0.73273, 0.7337
3, 0.73473, 0.73574, 0.73674, 0.73774, 0.73874, 0.73974, 0.74074,
0.74174, 0.74274, 0.74374,
0.74474, 0.74575, 0.74675, 0.74775, 0.74875, 0.74975, 0.75
075, 0.75175, 0.75275, 0.75375, 0.75475, 0.75576, 0.75676, 0.7577
6, 0.75876, 0.75976, 0.76076, 0.76176, 0.76276, 0.76376, 0.76476,
0.76577, 0.76677, 0.76777,
0.76877, 0.76977, 0.77077, 0.77177, 0.77277, 0.77377, 0.77
477, 0.77578, 0.77678, 0.77778, 0.77878, 0.77978, 0.78078, 0.7817
8, 0.78278, 0.78378, 0.78478, 0.78579, 0.78679, 0.78779, 0.78879,
0.78979, 0.79079, 0.79179,
0.79279, 0.79379, 0.79479, 0.7958, 0.7968, 0.7978, 0.7
988, 0.7998, 0.8008, 0.8018, 0.8028, 0.8038, 0.8048, 0.8058
1, 0.80681, 0.80781, 0.80881, 0.80981, 0.81081, 0.81181, 0.81281,
0.81381, 0.81481, 0.81582,
0.81682, 0.81782, 0.81882, 0.81982, 0.82082, 0.82182, 0.82
282, 0.82382, 0.82482, 0.82583, 0.82683, 0.82783, 0.82883, 0.8298
3, 0.83083, 0.83183, 0.83283, 0.83383, 0.83483, 0.83584, 0.83684,
0.83784, 0.83884, 0.83984,
0.84084, 0.84184, 0.84284, 0.84384, 0.84484, 0.84585, 0.84
685, 0.84785, 0.84885, 0.84985, 0.85085, 0.85185, 0.85285, 0.8538

```
5,     0.85485,     0.85586,     0.85686,     0.85786,     0.85886,     0.85986,     0.86086,
0.86186,     0.86286,     0.86386,
          0.86486,     0.86587,     0.86687,     0.86787,     0.86887,     0.86987,     0.87
087,     0.87187,     0.87287,     0.87387,     0.87487,     0.87588,     0.87688,     0.8778
8,     0.87888,     0.87988,     0.88088,     0.88188,     0.88288,     0.88388,     0.88488,
0.88589,     0.88689,     0.88789,
          0.88889,     0.88989,     0.89089,     0.89189,     0.89289,     0.89389,     0.89
489,      0.8959,      0.8969,      0.8979,      0.8989,      0.8999,      0.9009,      0.901
9,      0.9029,      0.9039,      0.9049,     0.90591,     0.90691,     0.90791,     0.90891,
0.90991,     0.91091,     0.91191,
          0.91291,     0.91391,     0.91491,     0.91592,     0.91692,     0.91792,     0.91
892,     0.91992,     0.92092,     0.92192,     0.92292,     0.92392,     0.92492,     0.9259
3,     0.92693,     0.92793,     0.92893,     0.92993,     0.93093,     0.93193,     0.93293,
0.93393,     0.93493,     0.93594,
          0.93694,     0.93794,     0.93894,     0.93994,     0.94094,     0.94194,     0.94
294,     0.94394,     0.94494,     0.94595,     0.94695,     0.94795,     0.94895,     0.9499
5,     0.95095,     0.95195,     0.95295,     0.95395,     0.95495,     0.95596,     0.95696,
0.95796,     0.95896,     0.95996,
          0.96096,     0.96196,     0.96296,     0.96396,     0.96496,     0.96597,     0.96
697,     0.96797,     0.96897,     0.96997,     0.97097,     0.97197,     0.97297,     0.9739
7,     0.97497,     0.97598,     0.97698,     0.97798,     0.97898,     0.97998,     0.98098,
0.98198,     0.98298,     0.98398,
          0.98498,     0.98599,     0.98699,     0.98799,     0.98899,     0.98999,     0.99
099,     0.99199,     0.99299,     0.99399,     0.99499,       0.996,       0.997,       0.99
8,       0.999,           1]), array([[    0.85714,     0.85714,     0.85714, ...,
0,           0,           0],
       [    0.80952,     0.80952,     0.80952, ...,           0,           0,           0],
       [       0.96,        0.96,        0.96, ...,           0,           0,           0]]),
'Confidence', 'Recall']]
fitness: 0.1376391831270174
keys: ['metrics/precision(B)', 'metrics/recall(B)', 'metrics/mAP50(B)', 'metrics/mAP50-95
(B)']
maps: array([    0.10532,    0.078458,     0.17549])
names: {0: 'Agricola', 1: 'Industria', 2: 'Poblacion'}
plot: True
results_dict: {'metrics/precision(B)': 0.36948484987296576, 'metrics/recall(B)': 0.3930158730
1587304, 'metrics/mAP50(B)': 0.2985665179364814, 'metrics/mAP50-95(B)': 0.11975836814818805,
'fitness': 0.1376391831270174}
save_dir: WindowsPath('runs/detect/val5')
speed: {'preprocess': 1.7213087815504808, 'inference': 55.0541510948768, 'loss': 0.0, 'postpr
ocess': 5.864400130051832}
task: 'detect'
```

Datos Temporales

In [ ]:

```python
import numpy as np
import rasterio
import os
from glob import glob
from skimage.transform import resize
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

IMAGENES_DIR = r"C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/images"
MASCARAS_DIR = r"C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/masks"
OUT_YOLO_DIR = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/npy_files/"

years = [2020, 2022, 2024]

for year in years:

    folder = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB_"+str(year)+"/"
```

```
    tif_files = sorted(glob("C:/Users/crome/Desktop/VIU/Code/Datos/AMB_"+str(year)+"/*.tif"))

    os.makedirs(IMAGENES_DIR, exist_ok=True)
    os.makedirs(MASCARAS_DIR, exist_ok=True)
    os.makedirs(OUT_YOLO_DIR, exist_ok=True)

    # u-net:
    target_rgb_shape = (256, 256, 3)
    target_mask_shape = (256, 256)

    errores = []

    for tif_path in tif_files:
        name = os.path.basename(tif_path).replace('.tif', '')

        try:
            with rasterio.open(tif_path) as src:
                img = src.read()

            rgb = np.stack([img[0], img[1], img[2]], axis=-1).astype(np.float32) / 10000.0
            mask = img[3].astype(np.uint8)

            # Redimensionar
            rgb = resize(rgb, target_rgb_shape, preserve_range=True, anti_aliasing=True)
            mask = resize(mask, target_mask_shape, order=0, preserve_range=True).astype(np.ui

            np.save(os.path.join(IMAGENES_DIR, f'{name}_rgb.npy'), rgb)
            np.save(os.path.join(MASCARAS_DIR, f'{name}_mask.npy'), mask)

        except Exception as e:
            print(f"Error procesando {name}: {e}")
            errores.append(name)

    print("Conversión completada.")
    if errores:
        print("Errores en archivos:", errores)

    # Yolo:

    for tif_file in sorted(glob(os.path.join(folder, 'AMB_*.tif'))):
        with rasterio.open(tif_file) as src:
            img = src.read([1, 2, 3])  # B4, B3, B2 (ajustar si es Sentinel)
            img = np.clip(img, 0, 10000) / 10000.0
            img = (img.transpose(1, 2, 0) * 255).astype(np.uint8)

            name = os.path.basename(tif_file).replace('.tif', '.jpg')
            Image.fromarray(img).save(os.path.join(OUT_YOLO_DIR, name))
```

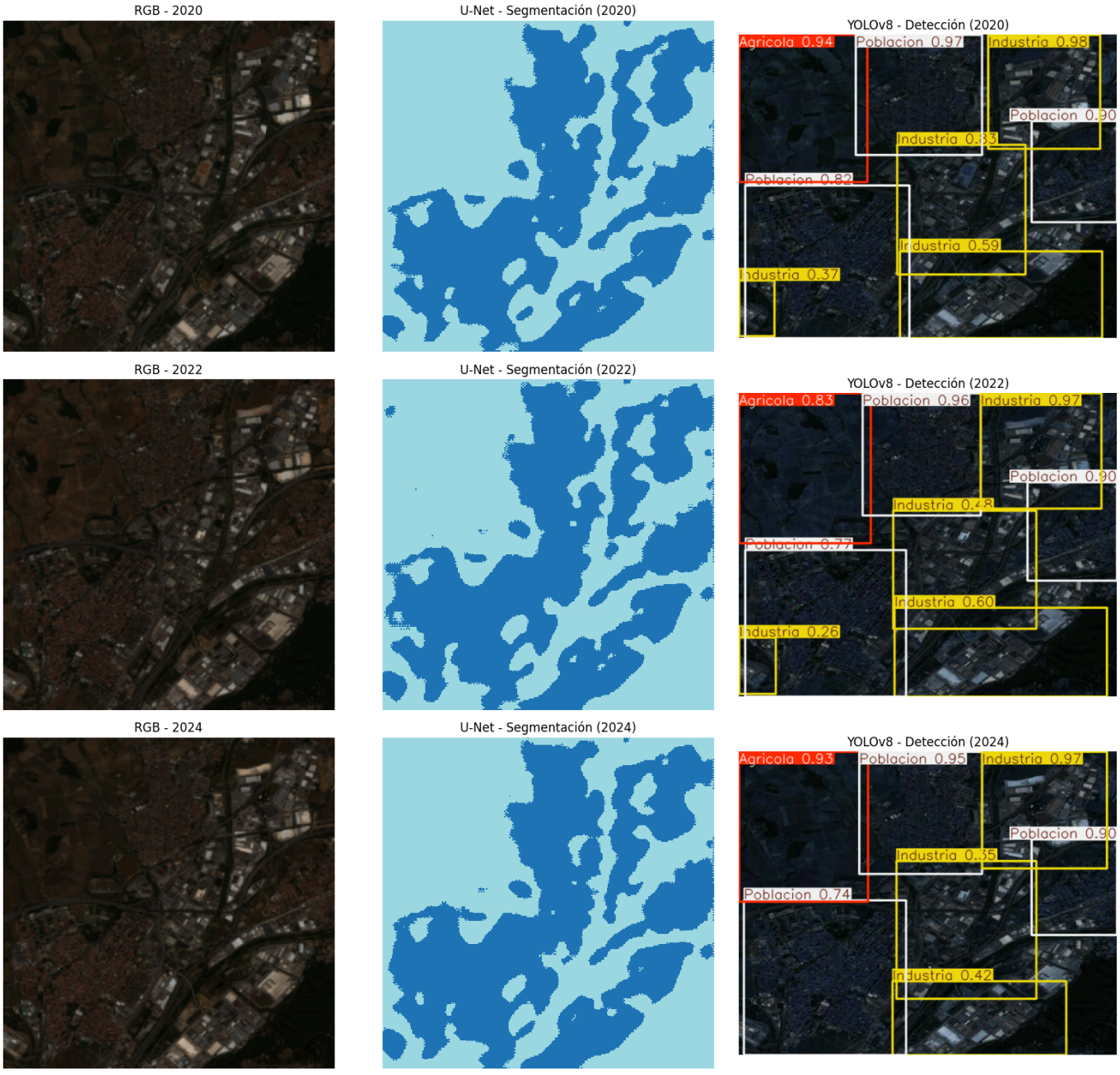
```
Conversión completada.
Conversión completada.
Conversión completada.
```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from ultralytics import YOLO
import os
import cv2
from PIL import Image

# Cargar modelos
model_unet = load_model("mejor_modelo.h5")
modelo_yolo = YOLO(r"C:\Users\crome\Desktop\VIU\Code\runs\detect\yolov8_model9\weights\best.pt


def predecir_unet(img):
```

```
    if img.max() > 1: img = img / 10000.0
    pred = model_unet.predict(np.expand_dims(img, axis=0))[0]
    return np.argmax(pred, axis=-1)

def predecir_yolo(imagen_path):
    resultados = modelo_yolo(imagen_path)
    return resultados[0]


fechas = ["2020", "2022", "2024"]
zona = "R7_C5"

# Ruta de imágenes .npy y .jpg
RGB_NPY_DIR = r"C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/images"
RGB_JPG_DIR = "C:/Users/crome/Desktop/VIU/Code/Datos/AMB_TEMP/npy_files/"

# Cargar imágenes .npy para U-Net
imagenes_rgb = {
    year: np.load(os.path.join(RGB_NPY_DIR, f"AMB_{zona}_{year}_rgb.npy"))
    for year in fechas
}

# Visualización conjunta
fig, axs = plt.subplots(len(fechas), 3, figsize=(16, 5 * len(fechas)))

for i, year in enumerate(fechas):
    # 1. Imagen RGB
    rgb = imagenes_rgb[year]

    # 2. Predicción U-Net
    pred_mask = predecir_unet(rgb)

    # 3. Predicción YOLO
    imagen_jpg_path = os.path.join(RGB_JPG_DIR, f"AMB_{zona}_{year}.jpg")
    pred_yolo = predecir_yolo(imagen_jpg_path)

    # Mostrar
    axs[i, 0].imshow(rgb)
    axs[i, 0].set_title(f"RGB - {year}")
    axs[i, 0].axis('off')

    axs[i, 1].imshow(pred_mask, cmap='tab20', interpolation='none')
    axs[i, 1].set_title(f"U-Net - Segmentación ({year})")
    axs[i, 1].axis('off')

    axs[i, 2].imshow(pred_yolo.plot())
    axs[i, 2].set_title(f"YOLOv8 - Detección ({year})")
    axs[i, 2].axis('off')

plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 1s 617ms/step

image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB_TEMP\npy_files\AMB_R7_C5_2020.jpg: 544x640
1 Agricola, 4 Industrias, 3 Poblacions, 316.4ms
Speed: 12.1ms preprocess, 316.4ms inference, 12.0ms postprocess per image at shape (1, 3, 544,
640)
1/1 [==============================] - 0s 184ms/step

image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB_TEMP\npy_files\AMB_R7_C5_2022.jpg: 544x640
1 Agricola, 4 Industrias, 3 Poblacions, 132.7ms
Speed: 8.1ms preprocess, 132.7ms inference, 0.0ms postprocess per image at shape (1, 3, 544, 6
40)
1/1 [==============================] - 0s 187ms/step

image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB_TEMP\npy_files\AMB_R7_C5_2024.jpg: 544x640
1 Agricola, 3 Industrias, 3 Poblacions, 210.6ms
Speed: 6.0ms preprocess, 210.6ms inference, 2.0ms postprocess per image at shape (1, 3, 544, 6
40)
```



In [2]:

```python
def comparar_segmentacion(mask1, mask2):
    cambio = mask1 != mask2
    total = cambio.size
    n_cambiado = np.sum(cambio)
    porcentaje_cambiado = 100 * n_cambiado / total
    return n_cambiado, porcentaje_cambiado

# Comparaciones entre años
```

```python
cambios_unet = {}
for y1, y2 in [("2020", "2022"), ("2022", "2024"), ("2020", "2024")]:
    m1 = predecir_unet(imagenes_rgb[y1])
    m2 = predecir_unet(imagenes_rgb[y2])
    n, p = comparar_segmentacion(m1, m2)
    cambios_unet[f"{y1}-{y2}"] = {"px_cambiados": n, "porcentaje": round(p, 2)}

# Mostrar resultados
print("Cambios en segmentación (U-Net):")
for k, v in cambios_unet.items():
    print(f"- {k}: {v['px_cambiados']} píxeles cambiados ({v['porcentaje']}%)")


from collections import Counter

# Inicializar diccionario
conteo_yolo = {}

for year in fechas:
    jpg_path = os.path.join(RGB_JPG_DIR, f"AMB_{zona}_{year}.jpg")
    resultado = predecir_yolo(jpg_path)

    clases = resultado.names
    ids_detectados = [int(cls) for cls in resultado.boxes.cls.cpu().numpy()]
    conteo = Counter(ids_detectados)

    # Guardamos con nombre de clase
    conteo_yolo[year] = {clases[k]: v for k, v in conteo.items()}

# Mostrar
print("Conteo YOLO por clase:")
for year in conteo_yolo:
    print(f"\n📅 Año {year}:")
    for clase, count in conteo_yolo[year].items():
        print(f"- {clase}: {count} detecciones")
```

```
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 357ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 113ms/step
Cambios en segmentación (U-Net):
- 2020-2022: 3016 píxeles cambiados (4.6%)
- 2022-2024: 2872 píxeles cambiados (4.38%)
- 2020-2024: 3284 píxeles cambiados (5.01%)

image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB_TEMP\npy_files\AMB_R7_C5_2020.jpg: 544x640
1 Agricola, 4 Industrias, 3 Poblacions, 92.2ms
Speed: 4.3ms preprocess, 92.2ms inference, 2.0ms postprocess per image at shape (1, 3, 544, 64
0)

image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB_TEMP\npy_files\AMB_R7_C5_2022.jpg: 544x640
1 Agricola, 4 Industrias, 3 Poblacions, 207.0ms
Speed: 4.2ms preprocess, 207.0ms inference, 2.0ms postprocess per image at shape (1, 3, 544, 6
40)

image 1/1 C:\Users\crome\Desktop\VIU\Code\Datos\AMB_TEMP\npy_files\AMB_R7_C5_2024.jpg: 544x640
1 Agricola, 3 Industrias, 3 Poblacions, 104.6ms
Speed: 2.0ms preprocess, 104.6ms inference, 0.0ms postprocess per image at shape (1, 3, 544, 6
40)
Conteo YOLO por clase:

🗓️ Año 2020:
- Industria: 4 detecciones
- Poblacion: 3 detecciones
- Agricola: 1 detecciones

🗓️ Año 2022:
- Industria: 4 detecciones
- Poblacion: 3 detecciones
- Agricola: 1 detecciones

🗓️ Año 2024:
- Industria: 3 detecciones
- Poblacion: 3 detecciones
- Agricola: 1 detecciones
```

In [6]:

```python
import matplotlib.pyplot as plt

def calcular_ict(unet_cambios, yolo_conteos, clases_objetivo=["Población", "Industria", "Agrí
    comparaciones = list(unet_cambios.keys())
    indice_compuesto = {}

    for comp in comparaciones:
        y1, y2 = comp.split("-")

        cambio_pct = unet_cambios[comp]["porcentaje"]
        cambio_norm = cambio_pct / 100

        yolo_y1 = sum(yolo_conteos[y1].get(cl, 0) for cl in clases_objetivo)
        yolo_y2 = sum(yolo_conteos[y2].get(cl, 0) for cl in clases_objetivo)
        yolo_diff = abs(yolo_y2 - yolo_y1)
        yolo_total = max(yolo_y1, yolo_y2, 1)
        yolo_norm = yolo_diff / yolo_total

        ictt = round((cambio_norm + yolo_norm) / 2, 3)

        indice_compuesto[comp] = {
            "cambio_pct": cambio_pct,
            "yolo_diff": yolo_diff,
```

```
            "cambio_norm": cambio_norm,
            "yolo_norm": yolo_norm,
            "ICTT": ictt
        }

    return indice_compuesto

# Uso:
ictt_resultado = calcular_ict(cambios_unet, conteo_yolo)

# Mostrar
print("\nÍndice Compuesto de Transformación Territorial (ICTT):")
for comp, v in ictt_resultado.items():
    print(f"- {comp}: {v['ICTT']} (U-Net: {v['cambio_pct']}%, YOLO Δ: {v['yolo_diff']})")
```

```
Índice Compuesto de Transformación Territorial (ICTT):
- 2020-2022: 0.023 (U-Net: 4.6%, YOLO Δ: 0)
- 2022-2024: 0.147 (U-Net: 4.38%, YOLO Δ: 1)
- 2020-2024: 0.15 (U-Net: 5.01%, YOLO Δ: 1)
```

In [7]:

```
# Visualizar como gráfico
comparaciones = list(ictt_resultado.keys())
unet_values = [ictt_resultado[c]["cambio_norm"] for c in comparaciones]
yolo_values = [ictt_resultado[c]["yolo_norm"] for c in comparaciones]
ictt_values = [ictt_resultado[c]["ICTT"] for c in comparaciones]

x = range(len(comparaciones))
width = 0.25

plt.figure(figsize=(10, 6))
plt.bar([i - width for i in x], unet_values, width=width, label="Cambio U-Net (normalizado)",
plt.bar(x, yolo_values, width=width, label="Cambio YOLO (normalizado)", color="#FF6F00")
plt.bar([i + width for i in x], ictt_values, width=width, label="ICTT (Índice Compuesto)", co

plt.xticks(x, comparaciones)
plt.ylabel("Valor normalizado")
plt.title("Índice Compuesto de Transformación Territorial (ICTT)")
plt.legend()
plt.grid(axis="y", linestyle="--", alpha=0.5)
plt.tight_layout()
plt.show()
```