YOLOv8 Pothole Detection and Road Damage Assessment

YOLOv8-seg, launched by **Ultralytics**, is accessible for easy installation via pip. Let's start by installing the Ultralytics package:

In [ ]:
```
# Install Ultralytics library
!pip install ultralytics
```

```
Collecting ultralytics
  Obtaining dependency information for ultralytics from https://files.pythonhosted.org/packages/3f/c8/b2ac54820cd983948475cd9b0
69168858c94f706d90695dbfdd97a807cbb/ultralytics-8.0.236-py3-none-any.whl.metadata
  Downloading ultralytics-8.0.236-py3-none-any.whl.metadata (35 kB)
Requirement already satisfied: matplotlib>=3.3.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (3.7.4)
Requirement already satisfied: numpy>=1.22.2 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (1.24.3)
Requirement already satisfied: opencv-python>=4.6.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (4.8.1.78)
Requirement already satisfied: pillow>=7.1.2 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (10.1.0)
Requirement already satisfied: pyyaml>=5.3.1 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (1.11.4)
Requirement already satisfied: torch>=1.8.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (2.0.0)
Requirement already satisfied: torchvision>=0.9.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (0.15.1)
Requirement already satisfied: tqdm>=4.64.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (4.66.1)
Requirement already satisfied: psutil in /opt/conda/lib/python3.10/site-packages (from ultralytics) (5.9.3)
Requirement already satisfied: py-cpuinfo in /opt/conda/lib/python3.10/site-packages (from ultralytics) (9.0.0)
Collecting thop>=0.1.1 (from ultralytics)
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
Requirement already satisfied: pandas>=1.1.4 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (2.0.3)
Requirement already satisfied: seaborn>=0.11.0 in /opt/conda/lib/python3.10/site-packages (from ultralytics) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultralytic
s) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultralytics)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultralytic
s) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultralytic
s) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultralytics)
(21.3)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultralytic
s) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=3.3.0->ultraly
tics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=1.1.4->ultralytics) (2023.
3)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python3.10/site-packages (from pandas>=1.1.4->ultralytics) (202
3.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests>=2.23.0->ultr
alytics) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests>=2.23.0->ultralytics) (3.
4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests>=2.23.0->ultralytic
s) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests>=2.23.0->ultralytic
s) (2023.11.17)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from torch>=1.8.0->ultralytics) (3.12.2)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.10/site-packages (from torch>=1.8.0->ultralytics)
(4.5.0)
Requirement already satisfied: sympy in /opt/conda/lib/python3.10/site-packages (from torch>=1.8.0->ultralytics) (1.12)
Requirement already satisfied: networkx in /opt/conda/lib/python3.10/site-packages (from torch>=1.8.0->ultralytics) (3.1)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages (from torch>=1.8.0->ultralytics) (3.1.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.7->matplotlib>=3.3.
0->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.10/site-packages (from jinja2->torch>=1.8.0->ultralyti
cs) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/lib/python3.10/site-packages (from sympy->torch>=1.8.0->ultralytics)
(1.3.0)
Downloading ultralytics-8.0.236-py3-none-any.whl (691 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 691.5/691.5 kB 23.4 MB/s eta 0:00:00
Installing collected packages: thop, ultralytics
Successfully installed thop-0.1.1.post2209072238 ultralytics-8.0.236
```

Next, I will import all necessary libraries required for our project:

In [ ]:
```
# Disable warnings in the notebook to maintain clean output cells
import warnings
```

```
warnings.filterwarnings('ignore')

# Import necessary libraries
import os
import shutil
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
import cv2
import yaml
from PIL import Image
from collections import deque
from ultralytics import YOLO
from IPython.display import Video
```

In [ ]:
```
# Configure the visual appearance of Seaborn plots
sns.set(rc={'axes.facecolor': '#ffe4de'}, style='darkgrid')
```

Step 2 | Loading YOLOv8-seg Pre-trained Model

Tabel of Contents

Here are the YOLOv8 pretrained Segment models. They come in various configurations, each optimized for speed and accuracy, and are pretrained on the COCO dataset, which includes 80 diverse object categories:

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

🚀 Choosing YOLOv8n-seg for Speed

The **YOLOv8-seg** lineup ranges from nano to xlarge, each striking a different balance between accuracy and speed. While larger models offer increased precision, they do so at the expense of response time. For our Pothole Segmentation task, which demands prompt processing, the **YOLOv8n-seg** is our chosen model. It's tailored for the fastest inference, making it highly suitable for real-time applications:

In [ ]:
```
# Load the pre-trained YOLOv8 nano segmentation model
model = YOLO('yolov8n-seg.pt')
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n-seg.pt to 'yolov8n-seg.pt'...
100%|████████| 6.73M/6.73M [00:00<00:00, 120MB/s]
```

After loading the YOLOv8 nano segmentation model, it's important to note that '**pothole**' is **not** one of the 80 object categories the model was initially trained to recognize in the COCO dataset. To identify potholes, we'll need to fine-tune the model on a specialized dataset of pothole images for precise segmentation.

Step 3 | Dataset Exploration

⬆️ Tabel of Contents

🛠️ Dataset Preparation for Model Fine-tuning

I've compiled a dataset specifically for training our pothole segmentation model, available on Kaggle. This dataset includes 780 images, with 720 for training and 60 for validation. Each image has been prepared to align correctly and resized to **640x640** pixels. To enhance the model's learning, the training images have undergone various augmentations, such as flipping, cropping, rotating, shearing, and adjusting brightness and exposure. This thorough preparation ensures our model will learn to identify and segment potholes effectively.

🧩 YOLOv8-seg Dataset Format

I've formatted our dataset specifically for the **YOLOv8-seg** model, prepared on Roboflow. It includes all elements essential for training a segmentation model effectively. Here is a breakdown:

1️⃣ **`train` and `valid` directories:**
Our dataset is split into `train` and `valid` directories, each with `images` and `labels` subdirectories. The `train` contains 720 image-label pairs, while `valid` has 60. Label files detail the class index and normalized coordinates for each pothole instance in the `[class-index] [x1] [y1] [x2] [y2] ... [xn] [yn]` format. In this format, `[class-index]` is the index of the class for the object, and `[x1] [y1] [x2] [y2] ... [xn] [yn]` are the bounding coordinates of the object's segmentation mask. The coordinates are separated by spaces.

2️⃣ **`data.yaml`:**
The `data.yaml` file which is the Ultralytics YOLO dataset configuration file. It specifies paths to the training and validation datasets, defines the number of classes (1), and the class name ('Pothole'). This format is crucial for setting up and training the model accurately with our dataset. Let's review the `data.yaml` to understand the setup:

In [ ]:
```yaml
# Define the dataset_path
dataset_path = '/kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8'

# Set the path to the YAML file
yaml_file_path = os.path.join(dataset_path, 'data.yaml')

# Load and print the contents of the YAML file
with open(yaml_file_path, 'r') as file:
    yaml_content = yaml.load(file, Loader=yaml.FullLoader)
    print(yaml.dump(yaml_content, default_flow_style=False))
```

```
names:
- Pothole
nc: 1
roboflow:
  license: CC BY 4.0
  project: pothole_segmentation_yolov8
  url: https://universe.roboflow.com/farzad/pothole_segmentation_yolov8/dataset/1
  version: 1
  workspace: farzad
train: ../train/images
val: ../valid/images
```

🗂️ Understanding the data.yaml File

The `data.yaml` file is crucial for configuring our model; it locates the training (`train: ../train/images`) and validation (`val: ../valid/images`) images and specifies that we're detecting a single class (`nc: 1`), **Pothole**. This setup ensures our **YOLOv8-seg** model is finely tuned to identify and segment potholes from our custom dataset. It's the guide that helps our model learn precisely what to look for during the segmentation task.

Now, let's continue our exploration by counting the images in both the training and validation sets and verifying their sizes:

In [ ]:
```python
# Set paths for training and validation image sets
train_images_path = os.path.join(dataset_path, 'train', 'images')
valid_images_path = os.path.join(dataset_path, 'valid', 'images')

# Initialize counters for the number of images
num_train_images = 0
num_valid_images = 0

# Initialize sets to hold the unique sizes of images
train_image_sizes = set()
valid_image_sizes = set()

# Check train images sizes and count
for filename in os.listdir(train_images_path):
    if filename.endswith('.jpg'):
        num_train_images += 1
        image_path = os.path.join(train_images_path, filename)
        with Image.open(image_path) as img:
            train_image_sizes.add(img.size)

# Check validation images sizes and count
for filename in os.listdir(valid_images_path):
    if filename.endswith('.jpg'):
        num_valid_images += 1
        image_path = os.path.join(valid_images_path, filename)
        with Image.open(image_path) as img:
            valid_image_sizes.add(img.size)

# Print the results
print(f"Number of training images: {num_train_images}")
print(f"Number of validation images: {num_valid_images}")

# Check if all images in training set have the same size
if len(train_image_sizes) == 1:
    print(f"All training images have the same size: {train_image_sizes.pop()}")
else:
    print("Training images have varying sizes.")

# Check if all images in validation set have the same size
if len(valid_image_sizes) == 1:
    print(f"All validation images have the same size: {valid_image_sizes.pop()}")
else:
    print("Validation images have varying sizes.")
```

```
Number of training images: 720
Number of validation images: 60
All training images have the same size: (640, 640)
All validation images have the same size: (640, 640)
```

📊 Dataset Analysis Insights

For our Pothole Segmentation project, the dataset comprises 720 training images and 60 validation images. All images are consistently sized at 640x640 pixels, matching the YOLOv8-seg model's input specifications. This uniformity ensures that the model trains and validates efficiently. The training to validation split provides a robust dataset for learning and a sufficient number for validation to assess the model's performance accurately.

Finally at this step, let's take a look at a few images from the dataset to get a sense of what the data looks like:

In [ ]:
```
# Set the seed for the random number generator
random.seed(0)

# Create a list of image files
image_files = [f for f in os.listdir(train_images_path) if f.endswith('.jpg')]

# Randomly select 15 images
random_images = random.sample(image_files, 15)

# Create a new figure
plt.figure(figsize=(19, 12))

# Loop through each image and display it in a 3x5 grid
for i, image_file in enumerate(random_images):
    image_path = os.path.join(train_images_path, image_file)
    image = Image.open(image_path)
    plt.subplot(3, 5, i + 1)
    plt.imshow(image)
    plt.axis('off')

# Add a suptitle
plt.suptitle('Random Selection of Dataset Images', fontsize=24)

# Show the plot
plt.tight_layout()
plt.show()

# Deleting unnecessary variable to free up memory
del image_files
```

Random Selection of Dataset Images

Step 4 | Fine-Tuning YOLOv8-seg

Tabel of Contents

In this step, I'll use transfer learning to fine-tune the **YOLOv8-seg model**, initially trained on the **COCO dataset**, for our Pothole Image Segmentation Dataset. Instead of starting the training from scratch with random weights, which requires a lot of data and time, I'll start with a model that already knows how to recognize various objects. This way, we save time and make the most out of our smaller dataset to teach the model how to identify and segment potholes in road images:

In [ ]:
```
# Train the model on our custom dataset
results = model.train(
    data=yaml_file_path,     # Path to the dataset configuration file
    epochs=150,              # Number of epochs to train for
    imgsz=640,               # Size of input images as integer
    patience=15,             # Epochs to wait for no observable improvement for early stopping of training
    batch=16,                # Number of images per batch
    optimizer='auto',        # Optimizer to use, choices=[SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto]
    lr0=0.0001,              # Initial learning rate
    lrf=0.01,                # Final learning rate (lr0 * lrf)
    dropout=0.25,            # Use dropout regularization
    device=0,                # Device to run on, i.e. cuda device=0
    seed=42                  # Random seed for reproducibility
)
```

```
Ultralytics YOLOv8.0.236 🚀 Python-3.10.12 torch-2.0.0 CUDA:0 (Tesla P100-PCIE-16GB, 16276MiB)
engine/trainer: task=segment, mode=train, model=yolov8n-seg.pt, data=/kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/data.yaml, epochs=150, time=None, patience=15, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=0, workers=8, project=None, name=train, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=42, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.25, val=True, split=val, save_json=False, save_hybrid=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_conf=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=False, opset=None, workspace=4, nms=False, lr0=0.0001, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_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, label_smoothing=0.0, 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, mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs/segment/train
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
```

```
100%|████████| 755k/755k [00:00<00:00, 22.1MB/s]
2024-01-07 17:59:26,072 INFO util.py:129 -- Outdated packages:
  ipywidgets==7.7.1 found, needs ipywidgets>=8
Run `pip install -U ipywidgets`, then restart the notebook server for rich notebook output.
2024-01-07 17:59:26,563 INFO util.py:129 -- Outdated packages:
  ipywidgets==7.7.1 found, needs ipywidgets>=8
Run `pip install -U ipywidgets`, then restart the notebook server for rich notebook output.
Overriding model.yaml nc=80 with nc=1

                   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   1004275  ultralytics.nn.modules.head.Segment          [1, 32, 64, [64, 128, 256]]
YOLOv8n-seg summary: 261 layers, 3263811 parameters, 3263795 gradients, 12.1 GFLOPs

Transferred 381/417 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/segment/train', view at http://localhost:6006/
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit:
  ......................................
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.16.1
Run data is saved locally in /kaggle/working/wandb/run-20240107_175942-xe4nnwyf
Syncing run train to Weights & Biases (docs)

View project at https://wandb.ai/nekouei-farzad/YOLOv8
View run at https://wandb.ai/nekouei-farzad/YOLOv8/runs/xe4nnwyf
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt to 'yolov8n.pt'...
100%|████████| 6.23M/6.23M [00:00<00:00, 94.7MB/s]
AMP: checks passed ✅
train: Scanning /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/train/labels... 720 images, 0 back
grounds, 0 corrupt: 100%|████████| 720/720 [00:01<00:00, 466.52it/s]
train: WARNING ⚠️ Cache directory /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/train is not wr
iteable, cache not saved.
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limi
t=(1, 4.0), tile_grid_size=(8, 8))
val: Scanning /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/labels... 60 images, 0 backgro
unds, 0 corrupt: 100%|████████| 60/60 [00:00<00:00, 450.75it/s]
val: WARNING ⚠️ Cache directory /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid is not writ
eable, cache not saved.
Plotting labels to runs/segment/train/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.0001' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'mome
ntum' automatically...
optimizer: AdamW(lr=0.002, momentum=0.9) with parameter groups 66 weight(decay=0.0), 77 weight(decay=0.0005), 76 bias(decay=0.
0)
150 epochs...

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
```

```
      1/150      3.11G       1.51      3.808       3.22      1.492         81        640:   2%|▏          | 1/45 [00:02<01:46,
2.42s/it]Exception in thread Thread-24 (plot_images):
Traceback (most recent call last):
  File "/opt/conda/lib/python3.10/threading.py", line 1016, in _bootstrap_inner
    self.run()
  File "/opt/conda/lib/python3.10/threading.py", line 953, in run
    self._target(*self._args, **self._kwargs)
  File "/opt/conda/lib/python3.10/site-packages/ultralytics/utils/plotting.py", line 637, in plot_images
    conf = confs[idx] if confs is not None else None  # check for confidence presence (label vs pred)
IndexError: The shape of the mask [81] at index 0 does not match the shape of the indexed tensor [16, 160, 160] at index 0
      1/150      3.11G      1.584      5.239      3.152      1.633         46        640:   4%|▍          | 2/45 [00:02<00:53,
1.25s/it]Exception in thread Thread-25 (plot_images):
Traceback (most recent call last):
  File "/opt/conda/lib/python3.10/threading.py", line 1016, in _bootstrap_inner
    self.run()
  File "/opt/conda/lib/python3.10/threading.py", line 953, in run
    self._target(*self._args, **self._kwargs)
  File "/opt/conda/lib/python3.10/site-packages/ultralytics/utils/plotting.py", line 637, in plot_images
    conf = confs[idx] if confs is not None else None  # check for confidence presence (label vs pred)
IndexError: The shape of the mask [46] at index 0 does not match the shape of the indexed tensor [16, 160, 160] at index 0
      1/150      3.11G      1.648      4.981       3.17      1.649         74        640:   7%|▋          | 3/45 [00:03<00:37,
1.13it/s]Exception in thread Thread-26 (plot_images):
Traceback (most recent call last):
  File "/opt/conda/lib/python3.10/threading.py", line 1016, in _bootstrap_inner
    self.run()
  File "/opt/conda/lib/python3.10/threading.py", line 953, in run
    self._target(*self._args, **self._kwargs)
  File "/opt/conda/lib/python3.10/site-packages/ultralytics/utils/plotting.py", line 637, in plot_images
    conf = confs[idx] if confs is not None else None  # check for confidence presence (label vs pred)
IndexError: The shape of the mask [74] at index 0 does not match the shape of the indexed tensor [16, 160, 160] at index 0
      1/150      3.12G      1.464      2.963      2.321      1.449         59        640: 100%|██████████| 45/45 [00:17<00:00,
2.56it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  2.77it/s]
                   all         60        201      0.735     0.0647      0.185     0.0733      0.735     0.0647      0.183
0.0885

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      2/150      3.08G      1.495      2.487      1.872      1.457         83        640: 100%|██████████| 45/45 [00:14<00:00,
3.13it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.19it/s]
                   all         60        201      0.139      0.373      0.148     0.0694      0.126      0.343      0.127
0.0503

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      3/150      3.32G      1.492      2.468      1.711       1.44         56        640: 100%|██████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.08it/s]
                   all         60        201      0.452      0.363      0.334      0.163      0.442      0.368      0.341
0.144

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      4/150      3.08G      1.524      2.435      1.676       1.45         82        640: 100%|██████████| 45/45 [00:14<00:00,
3.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.18it/s]
                   all         60        201      0.524      0.408      0.359      0.174      0.521        0.4      0.347
0.168

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      5/150      3.16G       1.43       2.32      1.522      1.409         59        640: 100%|██████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.40it/s]
                   all         60        201      0.467      0.498      0.419      0.196      0.411      0.453      0.365
0.153

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      6/150      3.08G      1.458      2.338      1.493       1.42         71        640: 100%|██████████| 45/45 [00:14<00:00,
3.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.61it/s]
                   all         60        201      0.597      0.478      0.525      0.305      0.649      0.487      0.542
0.285

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      7/150      3.15G      1.418      2.286      1.445      1.398         61        640: 100%|██████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.39it/s]
                   all         60        201      0.662      0.565       0.55      0.288      0.674      0.575      0.565
0.273

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      8/150      3.16G      1.427      2.256        1.4      1.393         69        640: 100%|██████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.39it/s]
```

```
                   all         60        201      0.586      0.552      0.569        0.3      0.586      0.552      0.559
0.299
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
      9/150      3.04G      1.374      2.215      1.356       1.37         67        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.53it/s]
                   all         60        201      0.638      0.592      0.603      0.297      0.667      0.587      0.614
0.303
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     10/150      3.11G      1.349      2.207      1.287      1.332         64        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.71it/s]
                   all         60        201      0.685      0.453      0.534      0.295      0.716      0.473      0.551
0.278
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     11/150      3.25G       1.35      2.179      1.235      1.341         63        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.62it/s]
                   all         60        201      0.678       0.45      0.537      0.288      0.674      0.463      0.533
0.261
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     12/150       3.1G      1.351      2.142      1.265      1.334         75        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.44it/s]
                   all         60        201      0.592      0.572      0.572      0.321      0.611      0.587      0.595
0.306
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     13/150      3.12G      1.351       2.12      1.249      1.339         71        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.48it/s]
                   all         60        201       0.66      0.642      0.639      0.361      0.666      0.647      0.649
0.342
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     14/150      3.16G      1.293      2.083      1.171      1.303         54        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.34it/s]
                   all         60        201      0.627      0.627      0.598       0.32      0.639      0.632      0.611
0.304
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     15/150      3.07G      1.325      2.089      1.174      1.309         59        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.74it/s]
                   all         60        201      0.636      0.532      0.553      0.299      0.631      0.527      0.549
0.293
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     16/150      3.04G       1.31      2.093      1.181      1.313         79        640: 100%|████████| 45/45 [00:14<00:00,
3.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.28it/s]
                   all         60        201      0.603      0.453      0.517      0.296      0.612      0.458       0.52
0.275
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     17/150      3.28G      1.258      2.024      1.121      1.273         77        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.47it/s]
                   all         60        201      0.685      0.652      0.676      0.387      0.702      0.632      0.671
0.376
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     18/150      3.28G      1.273      2.013      1.106      1.297         87        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.57it/s]
                   all         60        201      0.642      0.617      0.623      0.358      0.655      0.625      0.647
0.35
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     19/150      3.21G      1.239      2.056      1.108      1.271         47        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.30it/s]
                   all         60        201       0.56      0.542       0.54      0.309      0.549      0.522      0.516
0.274
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
```

```
     20/150      3.19G      1.245      1.979      1.062      1.267         70        640: 100%|██████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.55it/s]
                   all         60        201      0.635      0.498      0.557      0.301      0.605      0.542       0.57
0.288

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     21/150      3.19G      1.211      1.966      1.036      1.246         68        640: 100%|██████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.58it/s]
                   all         60        201      0.605      0.603      0.626      0.345       0.61      0.607      0.638
0.33

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     22/150      3.19G      1.207      1.929      1.021      1.249         63        640: 100%|██████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.50it/s]
                   all         60        201      0.647      0.647      0.659      0.371      0.701      0.662       0.68
0.369

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     23/150      3.64G      1.174       1.88      0.997      1.216         55        640: 100%|██████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.62it/s]
                   all         60        201       0.64      0.642      0.635      0.361      0.651      0.687       0.668
0.363

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     24/150      3.25G      1.199      1.923      1.041      1.244         62        640: 100%|██████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.62it/s]
                   all         60        201       0.69      0.575      0.637      0.367      0.707       0.59      0.654
0.347

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     25/150      3.23G      1.205      1.909     0.9997      1.234         77        640: 100%|██████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.65it/s]
                   all         60        201      0.623        0.6      0.611      0.326      0.629      0.607       0.618
0.323

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     26/150      3.15G      1.209      1.896      1.008      1.247         54        640: 100%|██████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.40it/s]
                   all         60        201      0.729      0.537      0.622      0.354      0.762      0.558       0.633
0.334

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     27/150       3.2G      1.155      1.886     0.9797      1.221         73        640: 100%|██████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.49it/s]
                   all         60        201      0.705      0.607      0.654      0.386      0.738      0.637       0.691
0.361

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     28/150      3.06G      1.167      1.863     0.9507      1.208         46        640: 100%|██████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.73it/s]
                   all         60        201      0.649      0.616      0.652      0.385      0.656      0.602       0.643
0.359

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     29/150       3.1G      1.172      1.869     0.9429      1.218         56        640: 100%|██████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.71it/s]
                   all         60        201      0.642      0.667      0.674      0.376      0.662      0.673       0.693
0.354

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     30/150      3.02G      1.163      1.878     0.9273      1.213         60        640: 100%|██████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.46it/s]
                   all         60        201      0.664      0.601      0.656      0.381      0.683      0.611       0.686
0.373

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     31/150      3.09G      1.145      1.813     0.9122       1.19         78        640: 100%|██████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:00<00:00,  3.67it/s]
```

```
                   all         60        201      0.575      0.632      0.605      0.341      0.656      0.552      0.598
0.332
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     32/150      3.01G      1.144      1.847     0.9383      1.199         68        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.72it/s]
                   all         60        201      0.615      0.652      0.665      0.387      0.628      0.697      0.689
0.385
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     33/150      3.34G      1.121      1.797     0.8855      1.184         85        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.18it/s]
                   all         60        201      0.665      0.612      0.615      0.353      0.685      0.627      0.639
0.328
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     34/150      3.06G      1.139      1.826     0.9098      1.197         74        640: 100%|████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.50it/s]
                   all         60        201      0.657      0.647      0.638      0.371      0.665      0.652      0.643
0.362
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     35/150      3.06G      1.106      1.808     0.8745      1.172         83        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.78it/s]
                   all         60        201      0.671      0.557      0.587      0.335      0.671      0.557       0.59
0.314
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     36/150      3.02G      1.144      1.814     0.8753      1.178         66        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.51it/s]
                   all         60        201      0.614      0.624      0.663      0.382      0.627      0.687      0.678
0.366
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     37/150      3.25G      1.092      1.789     0.8804      1.174         75        640: 100%|████████| 45/45 [00:14<00:00,
3.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.08it/s]
                   all         60        201      0.684      0.604      0.662      0.382      0.696      0.615      0.666
0.368
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     38/150       3.1G      1.094      1.759     0.8663      1.174         54        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.68it/s]
                   all         60        201      0.651      0.667      0.654      0.372      0.665      0.682      0.685
0.366
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     39/150      3.17G      1.091      1.724     0.8299      1.172         59        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.59it/s]
                   all         60        201      0.629      0.572       0.62      0.356      0.738      0.533       0.65
0.347
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     40/150      3.51G      1.084      1.755     0.8504       1.16        104        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.37it/s]
                   all         60        201      0.661       0.66      0.656      0.385      0.682      0.652      0.662
0.354
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     41/150      3.07G      1.082      1.739     0.8075      1.144         91        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.38it/s]
                   all         60        201      0.696      0.637      0.674      0.385      0.744      0.637      0.701
0.369
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     42/150      3.23G      1.067      1.702     0.8016      1.152         78        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.63it/s]
                   all         60        201      0.712      0.597      0.655      0.388       0.75      0.617      0.683
0.371
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
```

```
     43/150       3.1G      1.071       1.73     0.8221      1.147         64        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.36it/s]
                   all         60        201       0.65      0.647      0.656      0.393      0.622      0.716      0.683
0.39
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     44/150      3.15G      1.054      1.693     0.7761      1.147         74        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.61it/s]
                   all         60        201      0.668      0.652      0.653      0.389      0.701      0.667      0.674
0.382
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     45/150      3.29G       1.05      1.669     0.8086      1.149         66        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.76it/s]
                   all         60        201      0.654      0.627      0.658      0.382      0.666      0.653      0.684
0.362
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     46/150      3.12G      1.025       1.65     0.7915      1.135         52        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.69it/s]
                   all         60        201      0.611      0.602      0.628      0.369      0.675      0.587      0.662
0.357
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     47/150      3.08G      1.063      1.655     0.7979      1.138         81        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.70it/s]
                   all         60        201      0.685      0.592      0.674        0.4      0.629      0.667      0.695
0.392
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     48/150      3.17G       1.04      1.676     0.7665      1.126         75        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.68it/s]
                   all         60        201      0.662      0.627      0.637      0.369      0.682      0.617      0.664
0.366
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     49/150      3.19G      1.025       1.65     0.7626      1.126         85        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.75it/s]
                   all         60        201      0.709      0.662      0.728      0.391      0.766      0.647      0.746
0.404
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     50/150      3.12G      1.032      1.648     0.7737      1.132         56        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.20it/s]
                   all         60        201      0.718      0.617       0.67      0.402      0.736      0.632      0.696
0.389
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     51/150      3.21G      1.055      1.699     0.7718      1.122         53        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.77it/s]
                   all         60        201      0.708      0.615      0.648      0.383      0.719      0.625      0.664
0.379
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     52/150      3.09G      1.033      1.664     0.7724       1.14         75        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.68it/s]
                   all         60        201      0.653      0.667      0.689      0.403      0.656      0.662      0.707
0.398
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     53/150      3.18G      1.016      1.623     0.7442      1.121         60        640: 100%|████████| 45/45 [00:14<00:00,
3.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.72it/s]
                   all         60        201      0.667      0.592      0.626      0.361      0.682      0.602      0.648
0.364
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     54/150      3.15G          1      1.602     0.7174      1.107         71        640: 100%|████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.72it/s]
```

```
                 all         60        201      0.682      0.632      0.669      0.401      0.698      0.647      0.691
0.391

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     55/150      3.08G     0.9988      1.621     0.7221      1.101         75        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.78it/s]
                 all         60        201      0.807      0.602      0.678      0.382        0.8      0.597      0.674
0.374

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     56/150      3.17G     0.9991       1.63      0.733      1.099         97        640: 100%|████████| 45/45 [00:14<00:00,
3.13it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.53it/s]
                 all         60        201      0.716      0.628      0.698      0.396      0.722      0.633      0.704
0.387

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     57/150      3.16G     0.9857      1.567     0.7049        1.1         51        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.61it/s]
                 all         60        201      0.649      0.635      0.669      0.383      0.744      0.582      0.698
0.386

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     58/150      3.53G     0.9861      1.627     0.7353      1.111         70        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.75it/s]
                 all         60        201      0.692      0.649      0.679      0.406      0.708      0.664      0.706
0.398

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     59/150      3.08G     0.9632      1.584     0.7251      1.096         65        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.68it/s]
                 all         60        201      0.686      0.685      0.695      0.417      0.703      0.701      0.726
0.397

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     60/150      3.23G     0.9855      1.587     0.7114      1.089         51        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.76it/s]
                 all         60        201      0.682       0.66      0.677      0.405      0.687      0.665      0.679
0.384

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     61/150      3.15G     0.9664      1.564     0.7043      1.095         82        640: 100%|████████| 45/45 [00:13<00:00,
3.22it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.65it/s]
                 all         60        201      0.628      0.672       0.66      0.389      0.638      0.684      0.683
0.385

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     62/150       3.1G     0.9661       1.55     0.7035      1.098         65        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.56it/s]
                 all         60        201      0.734      0.607       0.67      0.412      0.732      0.622      0.687
0.39

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     63/150      3.09G     0.9732      1.565     0.7004      1.093         74        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.68it/s]
                 all         60        201      0.631      0.642      0.664      0.399      0.572      0.713      0.658
0.371

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     64/150      3.08G     0.9515      1.537     0.6891      1.087         74        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.29it/s]
                 all         60        201      0.676      0.652       0.67      0.416      0.687      0.662      0.693
0.394

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size

     65/150      3.08G     0.9462      1.515     0.6684      1.075         70        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.63it/s]
                 all         60        201      0.677      0.622      0.675      0.406      0.683      0.633      0.686
0.404

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
```

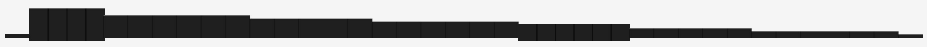
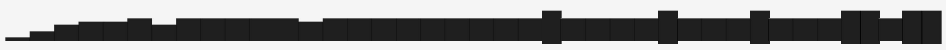
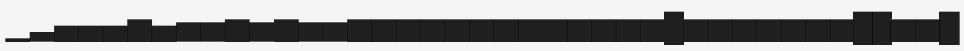
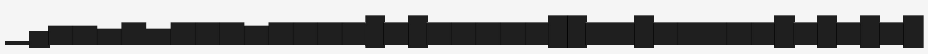
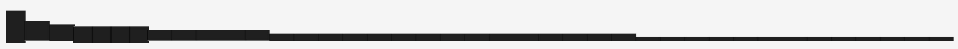
```
     66/150      3.15G     0.9705      1.571     0.6847      1.081         99        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.24it/s]
                   all         60        201      0.668       0.57      0.614      0.375      0.696      0.592       0.65
0.361
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     67/150      3.15G     0.9385      1.528     0.6728      1.068         64        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.82it/s]
                   all         60        201      0.689      0.682      0.696      0.433      0.705      0.711      0.719
0.407
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     68/150       3.1G     0.9421      1.534     0.6671      1.082         60        640: 100%|████████| 45/45 [00:14<00:00,
3.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.73it/s]
                   all         60        201      0.691       0.69      0.725      0.426      0.691       0.69      0.715
0.404
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     69/150      3.07G     0.9469       1.52     0.6697      1.083         58        640: 100%|████████| 45/45 [00:13<00:00,
3.23it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.06it/s]
                   all         60        201      0.649      0.673      0.668      0.399      0.645      0.706        0.7
0.405
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     70/150      3.12G     0.9297       1.55     0.6698      1.078         49        640: 100%|████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.44it/s]
                   all         60        201      0.743      0.574      0.666      0.416      0.756        0.6      0.703
0.39
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     71/150      3.18G      0.937      1.509     0.6645      1.065         98        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.71it/s]
                   all         60        201      0.725      0.612      0.672      0.393      0.732      0.617      0.668
0.386
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     72/150       3.1G     0.9064      1.514     0.6588      1.074        109        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.71it/s]
                   all         60        201      0.659      0.692      0.699      0.419      0.689      0.701      0.723
0.411
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     73/150      3.27G     0.9174      1.467     0.6517      1.054         77        640: 100%|████████| 45/45 [00:14<00:00,
3.19it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.71it/s]
                   all         60        201      0.695      0.632      0.666      0.392      0.709      0.619      0.671
0.395
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     74/150      3.33G      0.911      1.464     0.6646      1.083        108        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.77it/s]
                   all         60        201      0.698      0.667        0.7       0.41      0.703      0.672      0.714
0.417
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     75/150      3.15G      0.927      1.511     0.6508      1.067         95        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.66it/s]
                   all         60        201      0.708      0.667      0.707      0.441      0.719      0.667      0.722
0.42
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     76/150      3.23G     0.9078      1.468     0.6329       1.07         56        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.58it/s]
                   all         60        201      0.695      0.612       0.67      0.407      0.742      0.632      0.704
0.415
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     77/150      3.53G     0.9024      1.456     0.6275      1.055         58        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.78it/s]
```

```
                 all         60        201      0.723      0.597      0.679      0.395      0.745      0.617      0.715
0.39
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     78/150      3.14G     0.9007      1.442     0.6296      1.066         70        640: 100%|████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.54it/s]
                 all         60        201      0.775      0.583      0.696      0.411      0.772      0.639      0.727
0.416
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     79/150      3.15G     0.8749      1.469     0.6172      1.052         60        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.59it/s]
                 all         60        201      0.813      0.577      0.664      0.413      0.819      0.597       0.69
0.396
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     80/150       3.1G     0.9047      1.477      0.626       1.06         97        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.80it/s]
                 all         60        201       0.72      0.639      0.708      0.422      0.736      0.654      0.728
0.417
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     81/150      3.07G      0.885      1.445      0.617      1.051        105        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.66it/s]
                 all         60        201      0.717      0.577      0.649       0.41      0.744      0.602       0.69
0.406
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     82/150      3.08G     0.8927      1.446     0.6295      1.065         59        640: 100%|████████| 45/45 [00:14<00:00,
3.13it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.71it/s]
                 all         60        201      0.692      0.582       0.67      0.394      0.714      0.597      0.688
0.384
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     83/150      3.15G     0.8651      1.416     0.5856      1.024         78        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.81it/s]
                 all         60        201      0.611      0.672       0.67      0.405      0.643      0.667      0.703
0.405
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     84/150      3.53G     0.8548      1.426      0.587      1.044         64        640: 100%|████████| 45/45 [00:14<00:00,
3.11it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.72it/s]
                 all         60        201      0.758      0.617      0.724      0.439      0.708      0.657      0.721
0.427
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     85/150      3.14G     0.8593       1.42     0.5901      1.033         70        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.80it/s]
                 all         60        201       0.69      0.647      0.685      0.421      0.706      0.657      0.708
0.405
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     86/150      3.12G     0.8623      1.438      0.593      1.039         48        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.48it/s]
                 all         60        201      0.705      0.637       0.68      0.402      0.664      0.711      0.712
0.4
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     87/150      3.11G     0.8551      1.407      0.598      1.043         61        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.51it/s]
                 all         60        201       0.73      0.622      0.693      0.432      0.754      0.642      0.726
0.42
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     88/150      3.04G     0.8447      1.404     0.5911      1.032         69        640: 100%|████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.70it/s]
                 all         60        201      0.678      0.692      0.691      0.425      0.688      0.701      0.728
0.416
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
```

```
     89/150      3.06G     0.8692      1.376     0.5855      1.036         69        640: 100%|████████| 45/45 [00:13<00:00,
3.22it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.52it/s]
                   all         60        201      0.674      0.657      0.674      0.403      0.679      0.662      0.691
0.396
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     90/150      3.09G     0.8431      1.382     0.5773      1.033         67        640: 100%|████████| 45/45 [00:14<00:00,
3.16it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.66it/s]
                   all         60        201      0.667      0.672       0.68      0.419      0.677      0.682      0.709
0.396
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     91/150      3.12G     0.8559      1.371     0.5577      1.036         65        640: 100%|████████| 45/45 [00:14<00:00,
3.20it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.72it/s]
                   all         60        201      0.774      0.607      0.707      0.413       0.78      0.612       0.72
0.407
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     92/150      3.09G     0.8457      1.386     0.5924      1.032         73        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.66it/s]
                   all         60        201       0.65      0.647      0.681      0.414      0.675      0.672      0.713
0.404
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     93/150      3.17G     0.8492      1.376     0.5652      1.021         98        640: 100%|████████| 45/45 [00:14<00:00,
3.15it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.46it/s]
                   all         60        201      0.632      0.657      0.675      0.408      0.662      0.687        0.7
0.397
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     94/150      3.16G     0.8363      1.374     0.5662       1.02         88        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.58it/s]
                   all         60        201      0.672      0.662      0.662      0.389      0.704      0.652      0.695
0.387
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     95/150      3.08G     0.8333      1.376     0.5529      1.034         90        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.32it/s]
                   all         60        201      0.711      0.677      0.693      0.415      0.732      0.697      0.722
0.403
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     96/150      3.15G     0.8367        1.4     0.5641      1.034         92        640: 100%|████████| 45/45 [00:14<00:00,
3.17it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.73it/s]
                   all         60        201       0.73      0.642      0.702      0.408      0.743      0.652      0.702
0.4
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     97/150      3.49G     0.8378      1.351     0.5577      1.021         62        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.73it/s]
                   all         60        201       0.65      0.701      0.697      0.426      0.757      0.622      0.717
0.413
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     98/150      3.23G     0.7969      1.339     0.5462      1.003         59        640: 100%|████████| 45/45 [00:14<00:00,
3.18it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.75it/s]
                   all         60        201      0.669      0.647      0.661      0.411       0.69      0.667      0.685
0.394
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     99/150      3.05G     0.8247      1.353     0.5441       1.01         55        640: 100%|████████| 45/45 [00:14<00:00,
3.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|████████| 2/2 [00:00<00:00,  3.67it/s]
                   all         60        201      0.663      0.647       0.68      0.397      0.673      0.657      0.698
0.397
Stopping training early as no improvement observed in last 15 epochs. Best results observed at epoch 84, best model saved as be
st.pt.
To update EarlyStopping(patience=15) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStoppin
g.
```

```
99 epochs completed in 0.439 hours.
Optimizer stripped from runs/segment/train/weights/last.pt, 6.8MB
Optimizer stripped from runs/segment/train/weights/best.pt, 6.8MB

Validating runs/segment/train/weights/best.pt...
Ultralytics YOLOv8.0.236 🚀 Python-3.10.12 torch-2.0.0 CUDA:0 (Tesla P100-PCIE-16GB, 16276MiB)
YOLOv8n-seg summary (fused): 195 layers, 3258259 parameters, 0 gradients, 12.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 2/2 [00:01<00:00,  1.53it/s]
                   all         60        201      0.703      0.652      0.723      0.439      0.708      0.657      0.721
0.427
Speed: 1.8ms preprocess, 3.4ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to runs/segment/train
VBox(children=(Label(value='11.308 MB of 11.308 MB uploaded\r'), FloatProgress(value=1.0, max=1.0)))
```

Run history:

lr/pg0	
lr/pg1	
lr/pg2	
metrics/mAP50(B)	
metrics/mAP50(M)	
metrics/mAP50-95(B)	
metrics/mAP50-95(M)	
metrics/precision(B)	
metrics/precision(M)	
metrics/recall(B)	
metrics/recall(M)	
model/GFLOPs	
model/parameters	
model/speed_PyTorch(ms)	
train/box_loss	
train/cls_loss	
train/dfl_loss	
train/seg_loss	
val/box_loss	
val/cls_loss	
val/dfl_loss	
val/seg_loss	

Run summary:

lr/pg0	0.00072
lr/pg1	0.00072
lr/pg2	0.00072
metrics/mAP50(B)	0.72322
metrics/mAP50(M)	0.72061
metrics/mAP50-95(B)	0.43877
metrics/mAP50-95(M)	0.42662
metrics/precision(B)	0.70285
metrics/precision(M)	0.70821
metrics/recall(B)	0.65174
metrics/recall(M)	0.65672
model/GFLOPs	12.109
model/parameters	3263811
model/speed_PyTorch(ms)	4.709
train/box_loss	0.8247
train/cls_loss	0.54409
train/dfl_loss	1.01004
train/seg_loss	1.3532
val/box_loss	1.52651
val/cls_loss	1.20491

val/dfl_loss	1.49244
val/seg_loss	2.80548

View run **train** at: https://wandb.ai/nekouei-farzad/YOLOv8/runs/xe4nnwyf
Synced 6 W&B file(s), 24 media file(s), 9 artifact file(s) and 0 other file(s)
Find logs at: `./wandb/run-20240107_175942-xe4nnwyf/logs`

Step 5 | Model Performance Evaluation

Tabel of Contents

After training our pothole-detecting model, it created several output files. These files show different results and details from the training. Let's check out what files we have:

In [ ]:
```
# Define the path to the directory
post_training_files_path = '/kaggle/working/runs/segment/train'

# List the files in the directory
!ls {post_training_files_path}
```

```
BoxF1_curve.png
BoxPR_curve.png
BoxP_curve.png
BoxR_curve.png
MaskF1_curve.png
MaskPR_curve.png
MaskP_curve.png
MaskR_curve.png
args.yaml
confusion_matrix.png
confusion_matrix_normalized.png
events.out.tfevents.1704650378.8ad5ad308dd2.86.0
labels.jpg
labels_correlogram.jpg
results.csv
results.png
val_batch0_labels.jpg
val_batch0_pred.jpg
val_batch1_labels.jpg
val_batch1_pred.jpg
weights
```

📁 Training Output Files Explanations

After training our model, we have these output files:

- **BoxF1_curve.png, MaskF1_curve.png:** These images show the F1 score over different confidence thresholds for bounding box and mask predictions respectively, indicating the balance between precision and recall.
- **BoxP_curve.png, MaskP_curve.png:** These graphs display the precision of the bounding box and mask predictions of the model as confidence levels vary.
- **BoxPR_curve.png, MaskPR_curve.png:** These are the precision-recall curves for bounding box and mask predictions, useful for seeing the trade-off between precision and recall for different confidence thresholds.
- **BoxR_curve.png, MaskR_curve.png:** These images show how recall for bounding box and mask predictions changes with different confidence levels.
- **confusion_matrix.png:** Illustrates the model's performance by showing the true versus predicted classifications for bounding boxes.
- **confusion_matrix_normalized.png:** A normalized version of the confusion matrix that represents the proportion of each class predictions relative to the total number of predictions.
- **labels.jpg:** It displays data distributions for detected objects by the model.
- **labels_correlogram.jpg:** A correlogram that analyzes how different predicted labels correlate with each other.

- **results.csv:** This csv file captures a comprehensive set of performance metrics recorded at each epoch during the model's training process.
- **results.png:** A complex graph showing various training and validation losses (box, segmentation, classification, and distribution focal losses), as well as precision and recall metrics. The 'B' denotes bounding box and 'M' denotes mask predictions, including mean average precision (mAP) scores at different intersection over union (IoU) thresholds.
- **train_batch0.jpg, train_batch1.jpg, train_batch2.jpg:** Sample images from the training dataset batches with model predictions overlaid for visual inspection.
- **val_batch0_labels.jpg, val_batch1_labels.jpg:** Validation set images with the true labels overlaid for comparison against the model's predictions.
- **val_batch0_pred.jpg, val_batch1_pred.jpg:** Validation set images with the model's predictions overlaid for evaluation.
- **weights folder:** Contains the `best.pt` and `last.pt` files, which are the best and most recent weights of our trained model respectively**.**
- **args.yaml:** A configuration file that contains the hyperparameters and settings used during training.
- **events.out.tfevents.*:** A TensorBoard log file that stores the events like losses and metrics that occurred during training.

In the following, I will carry out a detailed examination and assessment of our model's performance, involving:

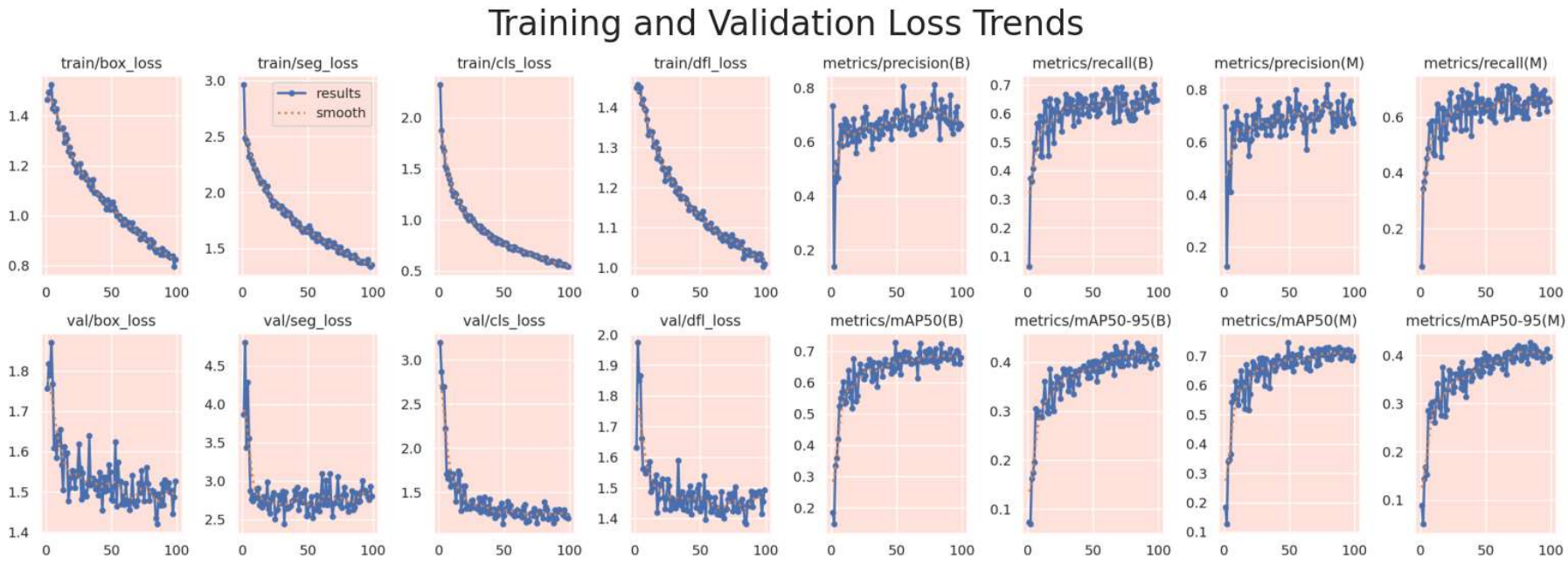
- **Learning Curves Analysis**
- **Confidence Threshold Metrics Analysis**
- **Precision-Recall Curve Analysis**
- **Confusion Matrix Analysis**
- **Validation Performance Metrics Assessment**

Step 5.1 | Learning Curves Analysis

🔝 Tabel of Contents

In this step, I'll examine the trends in training and validation losses over epochs to evaluate how well our segmentation model is learning. These loss trends are essential indicators of the model's learning progress and can reveal if the model is improving, overfitting, or underfitting. The `results.png` file captures these trends and will be our reference. Let's first dive into this graph:

In [ ]:
```python
# Create the full file path by joining the directory path with the filename
results_file_path = os.path.join(post_training_files_path, 'results.png')

# Read the image using cv2
image = cv2.imread(results_file_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image using matplotlib
plt.figure(figsize=(20, 8))
plt.imshow(image)
plt.title('Training and Validation Loss Trends', fontsize=24)
plt.axis('off')
plt.show()
```



📊 Learning Curves Analysis

The `results.png` file contains vital visual data tracking the model's learning progress across various metrics over epochs:

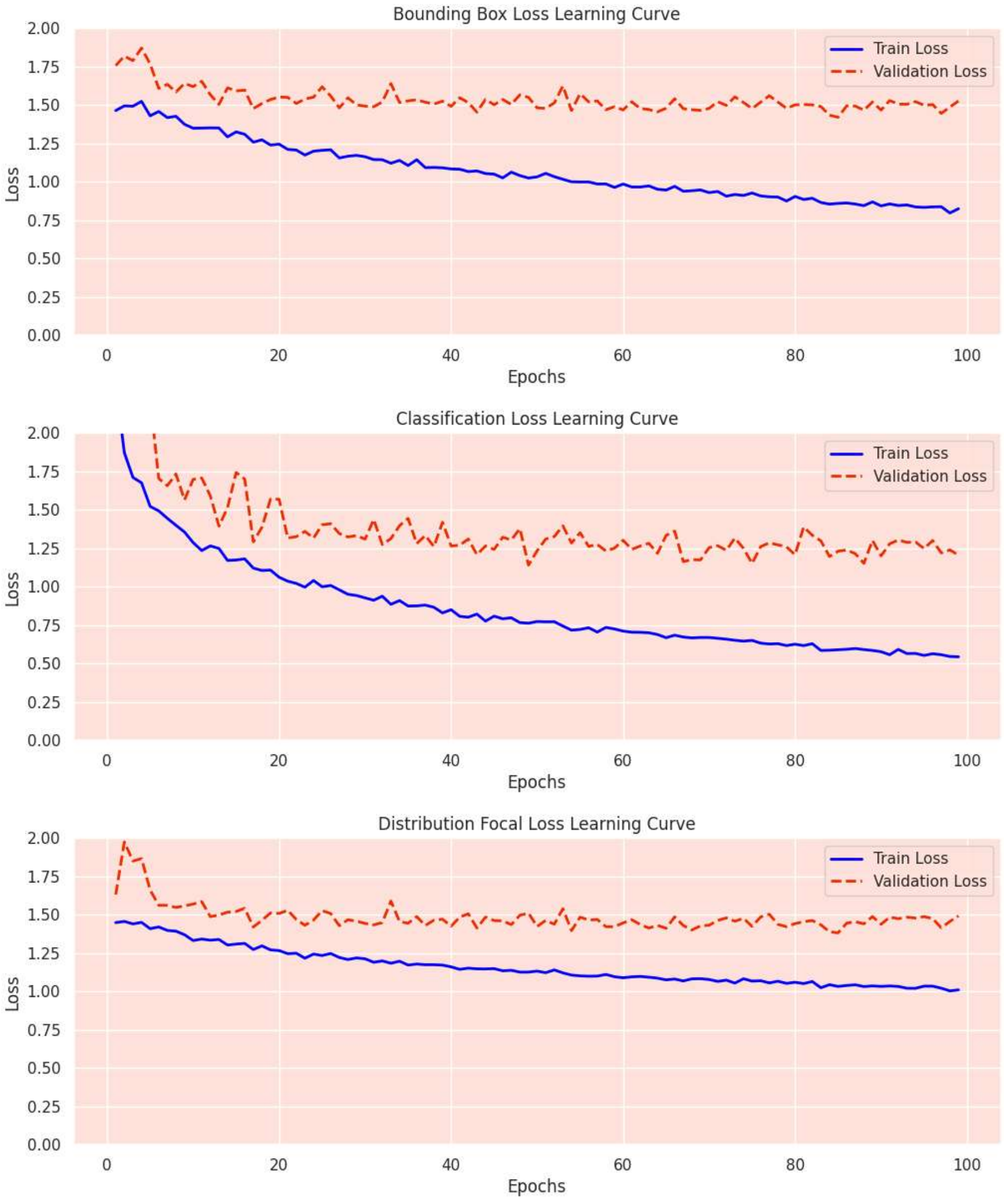
- **train/box_loss, val/box_loss:** These charts track the model's bounding box loss during training and validation, indicating how accurately the model is able to predict the boxes over time.
- **train/seg_loss, val/seg_loss:** Here, we see the segmentation loss, which shows the model's accuracy in segmenting the images throughout the training and validation phases.
- **train/cls_loss, val/cls_loss:** These plots reveal the classification loss, illustrating the model's ability to correctly classify the objects within the bounding boxes.
- **train/dfl_loss, val/dfl_loss:** These charts represent the distribution focal loss, a measure of how well the model is learning to differentiate and accurately classify various objects and segments in the images, particularly focusing on the challenging or hard-to-classify cases.
- **metrics/precision(B), metrics/precision(M):** Show precision metrics for bounding boxes (B) and masks (M), reflecting the proportion of correct positive predictions made by the model.
- **metrics/recall(B), metrics/recall(M):** Indicate recall metrics, which assess the model's ability to detect all relevant instances in the dataset.
- **metrics/mAP50(B), metrics/mAP50-95(B):** Mean Average Precision at IOU=0.50 and across IOU=0.50-0.95 for bounding box predictions, providing a single-figure summary of accuracy.
- **metrics/mAP50(M), metrics/mAP50-95(M):** Similar to the bounding box metrics, these measure the model's mask prediction accuracy at specific IOU thresholds.

To discern whether our model is progressing, overfitting, or underfitting, it's essential to examine the loss metrics — box_loss, seg_loss, cls_loss, dfl_loss — for both the training and validation datasets side by side. This comparison is made clearer by plotting the loss curves together. I'll utilize the `results.csv` file, which contains a detailed record of performance metrics at every epoch throughout the model's training, to redraw these informative curves:

In [ ]:
```python
# Define a function to plot learning curves for loss values
def plot_learning_curve(df, train_loss_col, val_loss_col, title, ylim_range=[0,2]):
    plt.figure(figsize=(12, 4))
    sns.lineplot(data=df, x='epoch', y=train_loss_col, label='Train Loss', color='blue', linestyle='-', linewidth=2)
    sns.lineplot(data=df, x='epoch', y=val_loss_col, label='Validation Loss', color='#ed2f00', linestyle='--', linewidth=2)
    plt.title(title)
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.ylim(ylim_range)
    plt.legend()
    plt.show()
```

In [ ]:
```python
# Create the full file path for 'results.csv' using the directory path and file name
results_csv_path = os.path.join(post_training_files_path, 'results.csv')
```

```python
# Load the CSV file from the constructed path into a pandas DataFrame
df = pd.read_csv(results_csv_path)

# Remove any leading whitespace from the column names
df.columns = df.columns.str.strip()

# Plot the learning curves for each loss
plot_learning_curve(df, 'train/box_loss', 'val/box_loss', 'Bounding Box Loss Learning Curve')
plot_learning_curve(df, 'train/cls_loss', 'val/cls_loss', 'Classification Loss Learning Curve')
plot_learning_curve(df, 'train/dfl_loss', 'val/dfl_loss', 'Distribution Focal Loss Learning Curve')
plot_learning_curve(df, 'train/seg_loss', 'val/seg_loss', 'Segmentation Loss Learning Curve', ylim_range=[0,5])
```



🔍 Model's Learning State

Our model's training progress, as shown by the learning curves, demonstrates a good fit on the validation data. The training and validation losses are close, which is a positive sign of the model's ability to generalize:

- **Bounding Box Loss:** The gap between training and validation loss suggests some overfitting, but the model is generally predicting boxes well.
- **Classification Loss:** Classification performance is strong on training data, with a slight drop on validation, indicating room for improvement.
- **Distribution Focal Loss:** Indicates focused learning on challenging examples, with validation performance suggesting further tuning may be beneficial.
- **Segmentation Loss:** Shows the model's capability in segmentation with some variability in validation loss, highlighting potential overfitting.

While it's challenging to completely eliminate overfitting, the close alignment of the curves is an encouraging indicator of generalization. **Adding more diverse data to our training set could further enhance the model's performance.**

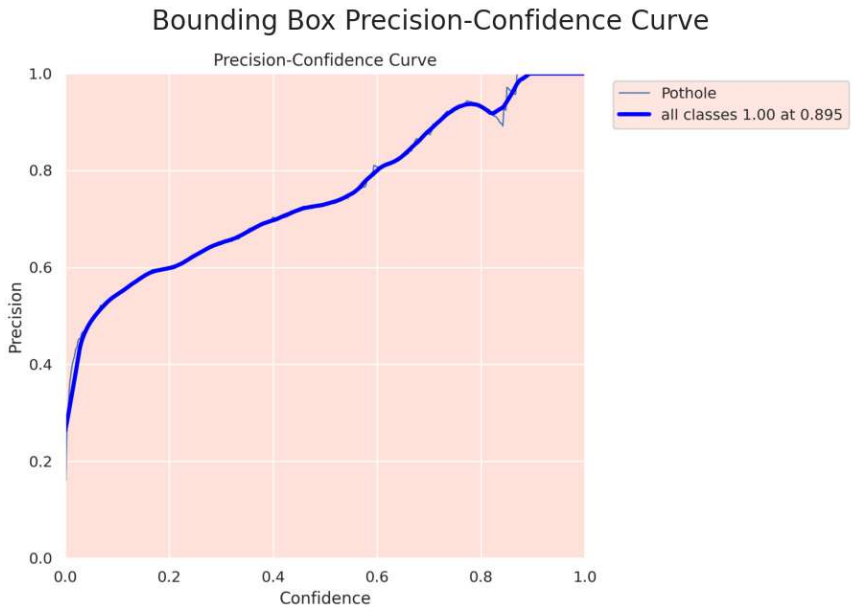
Step 5.2 | Confidence Threshold Metrics Analysis

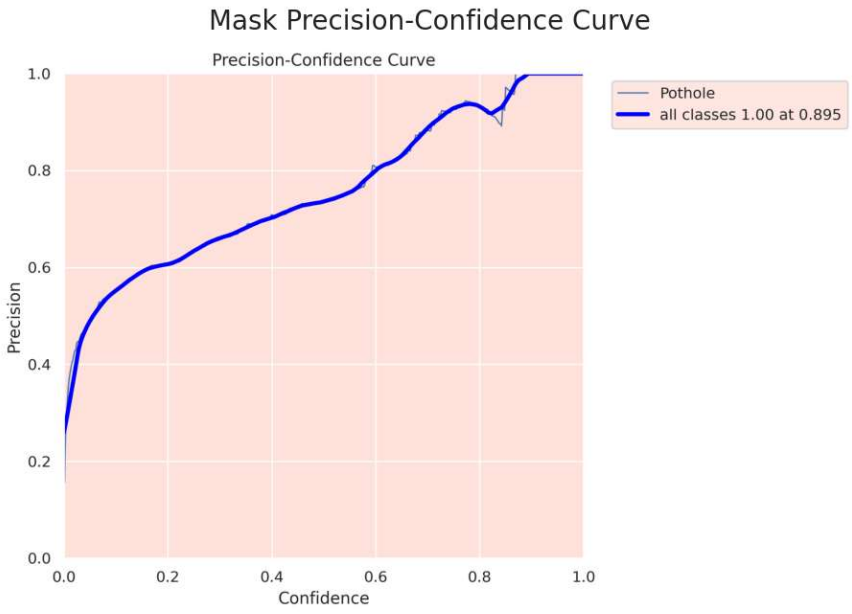
⬆ Tabel of Contents

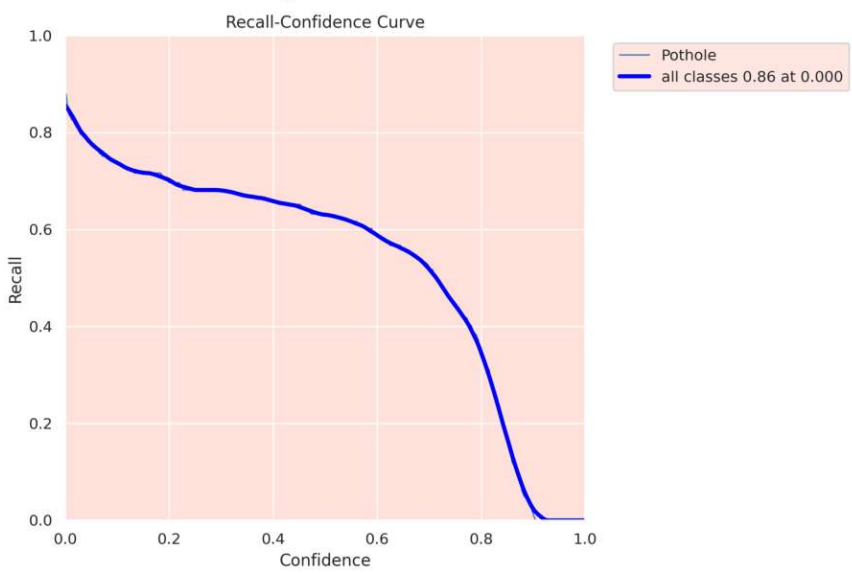
Now, I am going to focus on evaluating our model's predictive performance at various levels of confidence using precision, recall, and F1 score metrics for both bounding box and mask predictions:

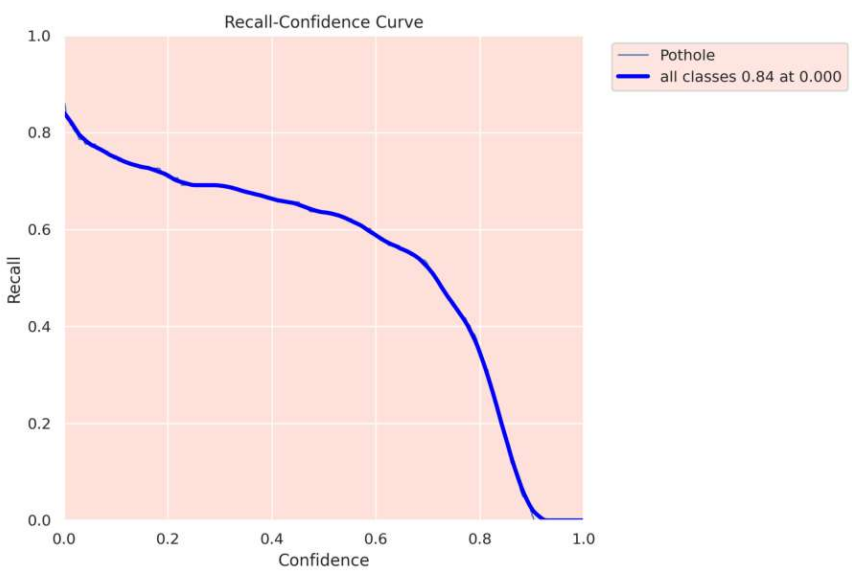
In [ ]:
```python
# Define the filenames for 'Box' and 'Mask' metrics along with their titles
box_files_titles = {
    'BoxP_curve.png': 'Bounding Box Precision-Confidence Curve',
    'BoxR_curve.png': 'Bounding Box Recall-Confidence Curve',
    'BoxF1_curve.png': 'Bounding Box F1-Confidence Curve'
}
mask_files_titles = {
    'MaskP_curve.png': 'Mask Precision-Confidence Curve',
    'MaskR_curve.png': 'Mask Recall-Confidence Curve',
    'MaskF1_curve.png': 'Mask F1-Confidence Curve'
}

# Create a 3x2 subplot
fig, axs = plt.subplots(3, 2, figsize=(20, 20))

# Function to read and convert image for plotting
def read_and_convert_image(file_path):
    # Read the image using cv2
    image = cv2.imread(file_path)
```

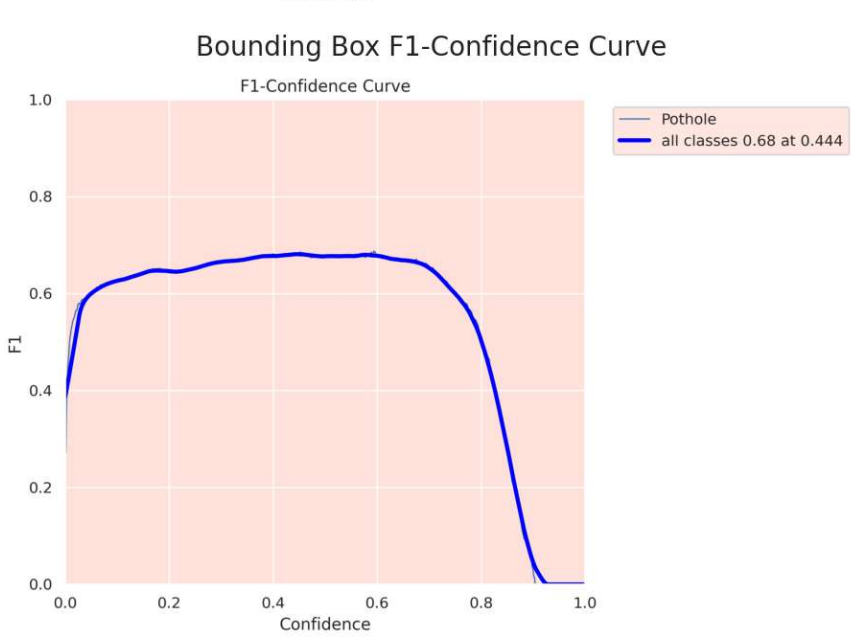
```
    # Convert from BGR to RGB
    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Plot 'Box' images in the first column with meaningful titles
for i, (filename, title) in enumerate(box_files_titles.items()):
    img_path = os.path.join(post_training_files_path, filename)
    img = read_and_convert_image(img_path)
    axs[i, 0].imshow(img)
    axs[i, 0].set_title(title, fontsize=20)
    axs[i, 0].axis('off')

# Plot 'Mask' images in the second column with meaningful titles
for i, (filename, title) in enumerate(mask_files_titles.items()):
    img_path = os.path.join(post_training_files_path, filename)
    img = read_and_convert_image(img_path)
    axs[i, 1].imshow(img)
    axs[i, 1].set_title(title, fontsize=20)
    axs[i, 1].axis('off')

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

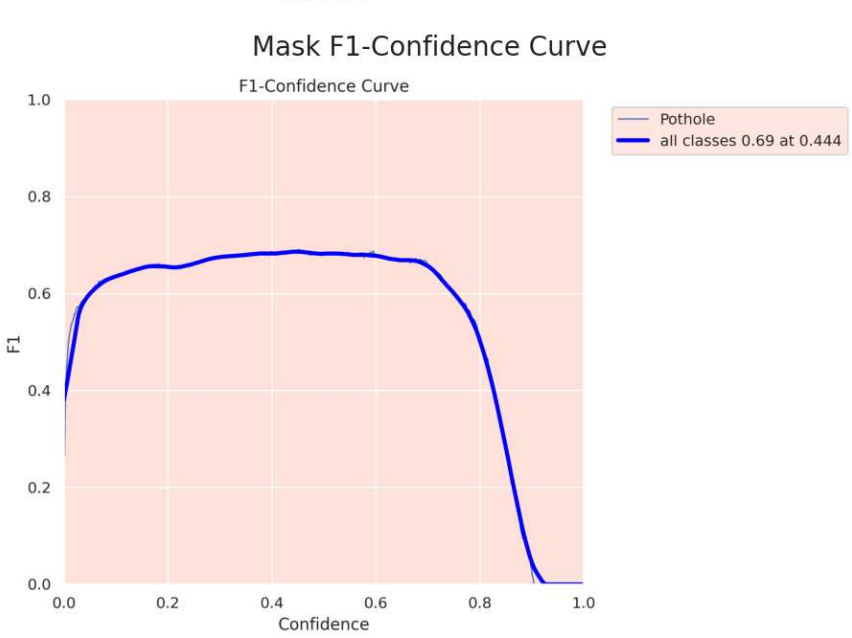












📈 Confidence Threshold Metrics Analysis

The precision, recall, and F1 score confidence curves demonstrate outstanding performance in our segmentation model:

- **Precision-Confidence Curve:** Shows near-perfect precision across all confidence levels, indicating very accurate predictions for both bounding boxes and masks.
- **Recall-Confidence Curve:** Maintains high recall across all confidence thresholds, suggesting the model consistently identifies true positives.
- **F1-Confidence Curve:** Reveals a stable and high F1 score, suggesting a balanced precision and recall, and the model performs well even at higher confidence thresholds.

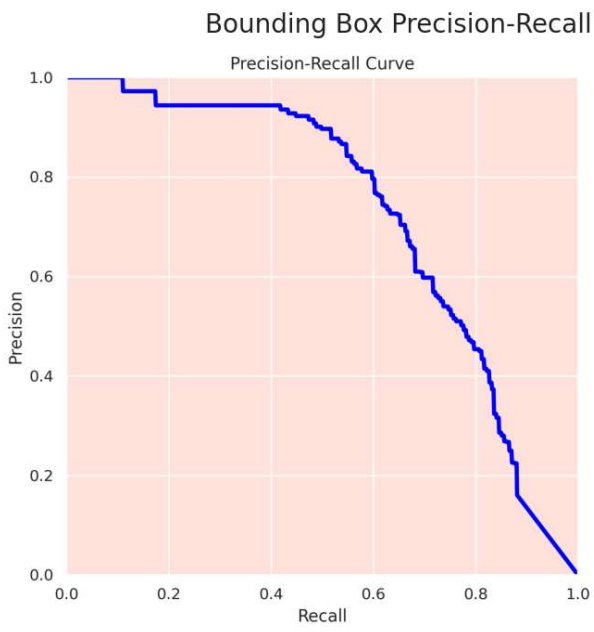
Overall, the model shows excellent generalization capabilities with robust predictive performance. The consistently high metrics across different confidence thresholds indicate that minimal adjustment is needed for practical application.

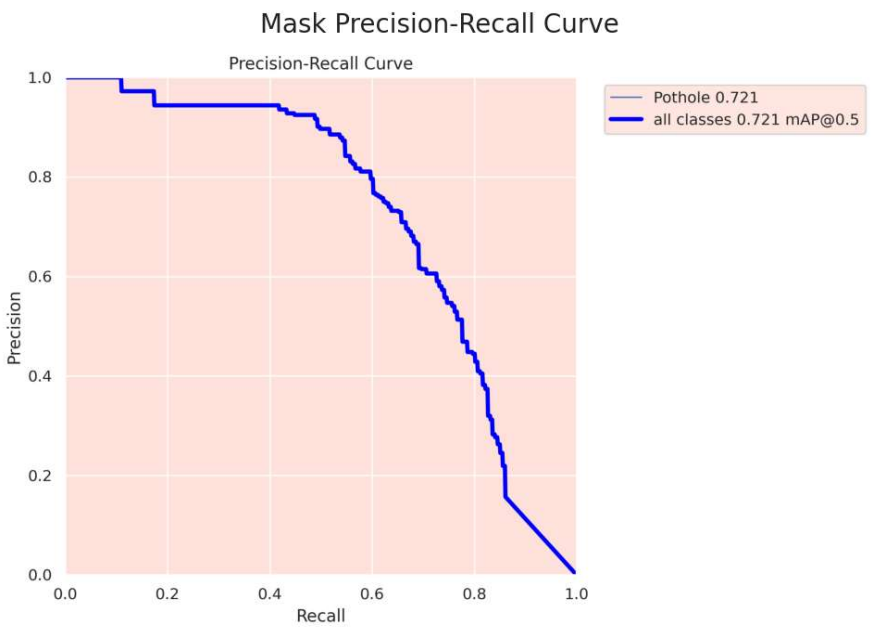
Step 5.3 | Precision-Recall Curve Analysis

Tabel of Contents

Now, I am going to analyze the precision-recall curves for both bounding box and mask predictions:

In [ ]:
```python
# Define the filenames for 'Box' and 'Mask' metrics along with their titles
pr_files_titles = {
    'BoxPR_curve.png': 'Bounding Box Precision-Recall Curve',
    'MaskPR_curve.png': 'Mask Precision-Recall Curve'
}

# Create a 1x2 subplot
fig, axs = plt.subplots(1, 2, figsize=(20, 10))

# Plot 'Box' and 'Mask' images in the subplot with meaningful titles
for i, (filename, title) in enumerate(pr_files_titles.items()):
    img_path = os.path.join(post_training_files_path, filename)
    img = read_and_convert_image(img_path)
    axs[i].imshow(img)
    axs[i].set_title(title, fontsize=20)
    axs[i].axis('off')

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





📈 Precision-Recall Curve Analysis

An examination of the precision-recall curves reveals key aspects of our segmentation model's performance:

- **Bounding Box Precision-Recall Curve:** The curve indicates a high level of precision, with a mean Average Precision (mAP) of 0.72, showing the model's effectiveness in accurately identifying objects.
- **Mask Precision-Recall Curve:** This curve suggests almost the same precision in segmenting objects, evidenced by a mAP of 0.72, pointing to the model's proficiency in mask predictions.

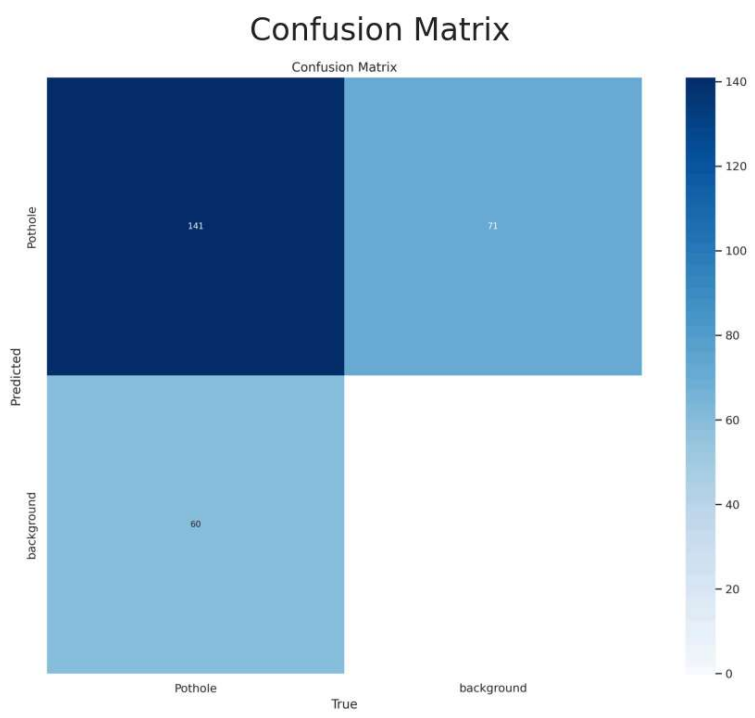
The high precision across various recall levels for both bounding boxes and masks indicates that the model performs reliably and can be trusted for practical use.

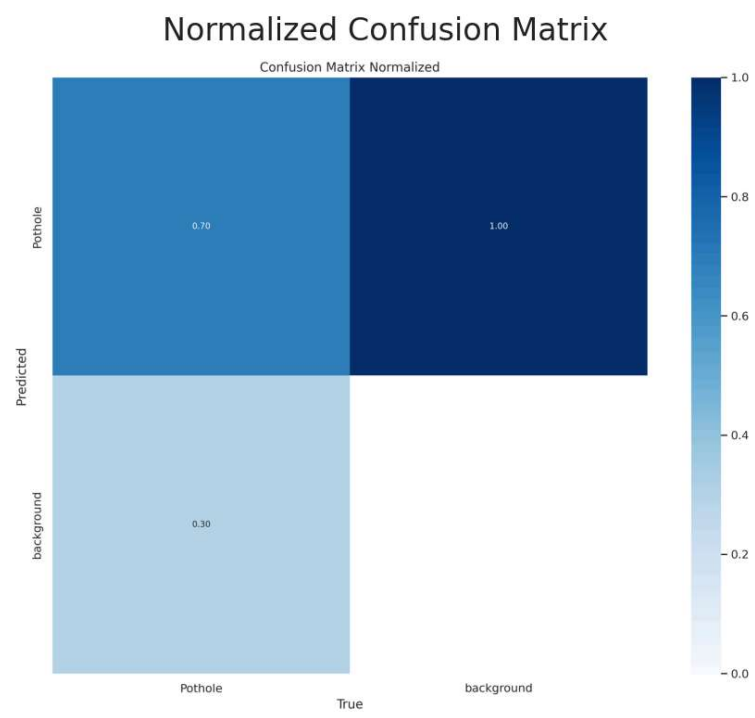
Step 5.4 | Confusion Matrix Analysis

Tabel of Contents

Next, I will concentrate on presenting and thoroughly examining the confusion matrix that reflects our model's predictive results on the validation set:

In [ ]:
```
# Construct the path to the confusion matrix images
confusion_matrix_path = os.path.join(post_training_files_path, 'confusion_matrix.png')
confusion_matrix_normalized_path = os.path.join(post_training_files_path, 'confusion_matrix_normalized.png')

# Create a 1x2 subplot
fig, axs = plt.subplots(1, 2, figsize=(20, 10))

# Read and convert both images
cm_img = read_and_convert_image(confusion_matrix_path)
cm_norm_img = read_and_convert_image(confusion_matrix_normalized_path)

# Display the images
axs[0].imshow(cm_img)
axs[0].set_title('Confusion Matrix', fontsize=24)
axs[0].axis('off')

axs[1].imshow(cm_norm_img)
axs[1].set_title('Normalized Confusion Matrix', fontsize=24)
axs[1].axis('off')

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





🔍 Confusion Matrix Analysis

The normalized confusion matrix for our YOLOv8 pothole segmentation model provides valuable insights into its performance. The model has a 70% true positive rate, meaning it correctly identifies potholes 70% of the time. However, there is a 30% false negative rate, indicating that some potholes are missed during detection. Given the challenging nature of pothole segmentation, this performance is commendable.

Step 5.5 | Validation Performance Metrics Assessment

Tabel of Contents

Finally, I am delving into various metrics to assess our best segmentation model's predictive capabilities on the validation set:

In [ ]:
```
# Construct the path to the best model weights file using os.path.join
best_model_path = os.path.join(post_training_files_path, 'weights/best.pt')

# Load the best model weights into the YOLO model
best_model = YOLO(best_model_path)

# Validate the best model using the validation set with default parameters
metrics = best_model.val(split='val')
```

```
Ultralytics YOLOv8.0.236 🚀 Python-3.10.12 torch-2.0.0 CUDA:0 (Tesla P100-PCIE-16GB, 16276MiB)
YOLOv8n-seg summary (fused): 195 layers, 3258259 parameters, 0 gradients, 12.0 GFLOPs
val: Scanning /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/labels... 60 images, 0 backgro
unds, 0 corrupt: 100%|██████████| 60/60 [00:00<00:00, 437.35it/s]
val: WARNING ⚠️ Cache directory /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid is not writ
eable, cache not saved.
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP5
0-95): 100%|██████████| 4/4 [00:04<00:00,  1.02s/it]
                   all         60        201      0.751      0.617      0.724       0.44      0.708      0.657      0.721
0.427
Speed: 4.4ms preprocess, 15.9ms inference, 0.0ms loss, 3.3ms postprocess per image
Results saved to runs/segment/val
```

In [ ]:
```
# Convert the dictionary to a pandas DataFrame and use the keys as the index
metrics_df = pd.DataFrame.from_dict(metrics.results_dict, orient='index', columns=['Metric Value'])

# Display the DataFrame
metrics_df.round(3)
```

Out[ ]:

	Metric Value
metrics/precision(B)	0.751
metrics/recall(B)	0.617
metrics/mAP50(B)	0.724
metrics/mAP50-95(B)	0.440
metrics/precision(M)	0.708
metrics/recall(M)	0.657
metrics/mAP50(M)	0.721
metrics/mAP50-95(M)	0.427
fitness	0.924

📊 Validation Performance Metrics Assessment

Our YOLOv8 **tiny** model for pothole segmentation demonstrates promising results in validation. Precision metrics for bounding boxes (B) and masks (M) show high values of 0.75 and 0.71, respectively, indicating accurate predictions. Recall scores are slightly lower, with 0.61 for bounding boxes and 0.66 for masks, suggesting some true potholes may be missed. The mAP50 scores are robust at 0.72 for both bounding box and mask predictions, but the mAP50-95 scores show there is room to improve the consistency of predictions across different IoU thresholds. The overall fitness score of 0.92 reflects a well-trained model with potential for fine-tuning.

Step 6 | Model Inference

⬆️ Tabel of Contents

To effectively evaluate how well our model performs on new data, I'll carry out inferences in two key stages:

- **Inference on Validation Images**
- **Inference on a New Test Video**

Step 6.1 | Inference on Validation Images

Tabel of Contents

To begin, I will showcase the capabilities of our best segmentation model on a selection of images from the validation set:

In [ ]:
```python
# Define the path to the validation images
valid_images_path = os.path.join(dataset_path, 'valid', 'images')

# List all jpg images in the directory
image_files = [file for file in os.listdir(valid_images_path) if file.endswith('.jpg')]

# Select 9 images at equal intervals
num_images = len(image_files)
selected_images = [image_files[i] for i in range(0, num_images, num_images // 9)]

# Initialize the subplot
fig, axes = plt.subplots(3, 3, figsize=(20, 21))
fig.suptitle('Validation Set Inferences', fontsize=24)

# Perform inference on each selected image and display it
for i, ax in enumerate(axes.flatten()):
    image_path = os.path.join(valid_images_path, selected_images[i])
    results = best_model.predict(source=image_path, imgsz=640)
    annotated_image = results[0].plot()
    annotated_image_rgb = cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB)
    ax.imshow(annotated_image_rgb)
    ax.axis('off')

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

```
image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-17-_jpg.rf.0d172b6acced
f4c52a3868d9b690d48b.jpg: 640x640 1 Pothole, 6.4ms
Speed: 1.4ms preprocess, 6.4ms inference, 8.6ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-262-_jpg.rf.02970860794
f859699c9f743bc0fd7d1.jpg: 640x640 5 Potholes, 6.5ms
Speed: 1.4ms preprocess, 6.5ms inference, 1.7ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-60-_jpg.rf.4e715308cc8d
c7455b767414cda028ab.jpg: 640x640 2 Potholes, 6.6ms
Speed: 1.3ms preprocess, 6.6ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-178-_jpg.rf.2e27d006aec
1544f1eec88d9170fd6ce.jpg: 640x640 1 Pothole, 6.4ms
Speed: 1.3ms preprocess, 6.4ms inference, 1.9ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-72-_jpg.rf.82c8314917ff
5284106bd3428cff5792.jpg: 640x640 1 Pothole, 6.4ms
Speed: 1.3ms preprocess, 6.4ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-33-_jpg.rf.143c56d136e3
245555b26b4d5108a4e1.jpg: 640x640 4 Potholes, 6.4ms
Speed: 1.4ms preprocess, 6.4ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-69-_jpg.rf.0fcb6cd22beb
140d931a387036b7eab6.jpg: 640x640 10 Potholes, 6.8ms
Speed: 1.3ms preprocess, 6.8ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-144-_jpg.rf.61a3b886f05
8ed3a4788426cfa7c4988.jpg: 640x640 2 Potholes, 6.5ms
Speed: 1.3ms preprocess, 6.5ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)

image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-36-_jpg.rf.13a668c40036
81c4631508465eef5a9f.jpg: 640x640 5 Potholes, 6.5ms
Speed: 1.3ms preprocess, 6.5ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)
```

Validation Set Inferences

Step 6.2 | Inference on a New Test Video

Tabel of Contents

Next, we'll see how well our model can handle a brand new test video it hasn't seen before. This is an important test to check if our model can be used in the real world, showing that it can do a good job even with new and different data:

In [ ]:
```
# Define the path to the sample video in the dataset
dataset_video_path = '/kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/sample_video.mp4'

# Define the destination path in the working directory
video_path = '/kaggle/working/sample_video.mp4'

# Copy the video file from its original location in the dataset to the current working directory in Kaggle
shutil.copyfile(dataset_video_path, video_path)

# Initiate vehicle detection on the sample video using the best performing model and save the output
best_model.predict(source=video_path, save=True)
```

To display the processed video in the notebook environment, I will convert the output `.avi` file into the widely supported `.mp4` format for better compatibility:

In [ ]:
```
# Convert the .avi video generated by the YOLOv8 prediction to .mp4 format for compatibility with notebook display
!ffmpeg -y -loglevel panic -i /kaggle/working/runs/segment/predict/sample_video.avi processed_sample_video.mp4

# Embed and display the processed sample video within the notebook
Video("processed_sample_video.mp4", embed=True, width=960)
```

Out[ ]:

After confirming our model's generalization capabilities, let's proceed to save the model with its best-performing weights:

In [ ]:
```
# Export the model
best_model.export(format='onnx')
```

```
Ultralytics YOLOv8.0.236 🚀 Python-3.10.12 torch-2.0.0 CPU (Intel Xeon 2.00GHz)

PyTorch: starting from '/kaggle/working/runs/segment/train/weights/best.pt' with input shape (1, 3, 640, 640) BCHW and output s
hape(s) ((1, 37, 8400), (1, 32, 160, 160)) (6.5 MB)

ONNX: starting export with onnx 1.15.0 opset 17...
================ Diagnostic Run torch.onnx.export version 2.0.0 ================
verbose: False, log level: Level.ERROR
======================== 0 NONE 0 NOTE 0 WARNING 0 ERROR ========================

ONNX: export success ✅ 0.9s, saved as '/kaggle/working/runs/segment/train/weights/best.onnx' (12.6 MB)

Export complete (2.5s)
Results saved to /kaggle/working/runs/segment/train/weights
Predict:         yolo predict task=segment model=/kaggle/working/runs/segment/train/weights/best.onnx imgsz=640
Validate:        yolo val task=segment model=/kaggle/working/runs/segment/train/weights/best.onnx imgsz=640 data=/kaggle/input/
pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/data.yaml
Visualize:       https://netron.app
```

Out[ ]: `'/kaggle/working/runs/segment/train/weights/best.onnx'`

Step 7 | Real-Time Road Damage Assessment

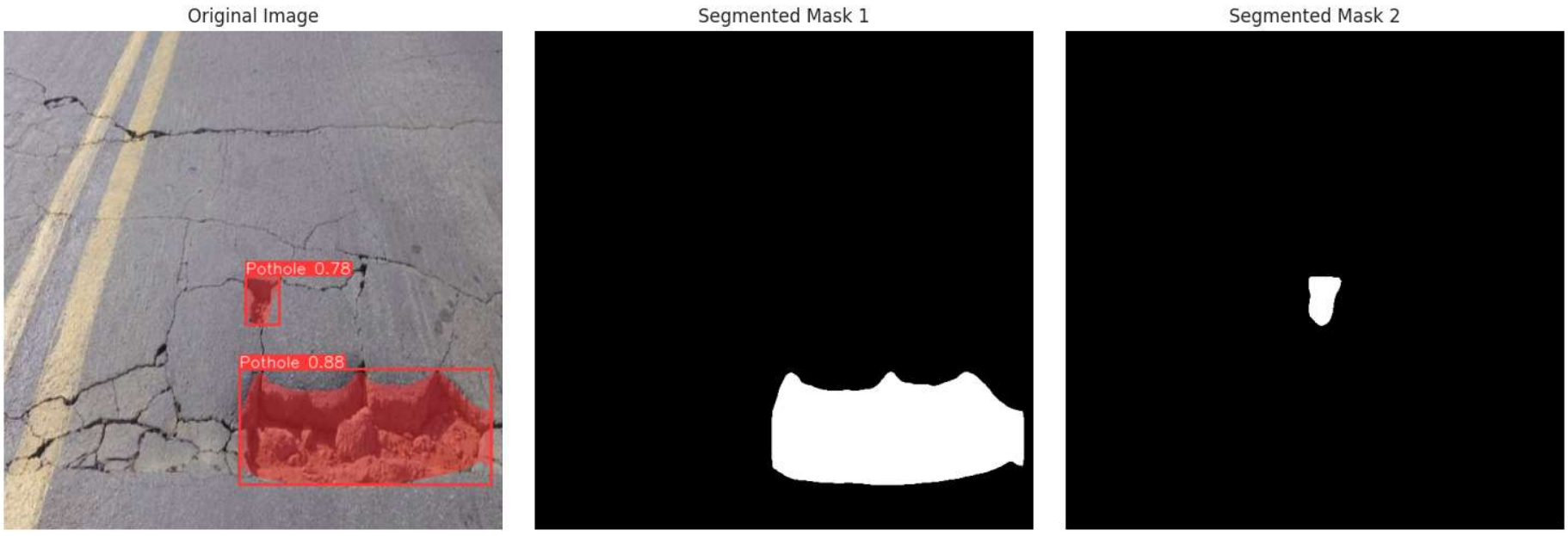
Tabel of Contents

As we move to the practical step of our project, I aim to deploy our final pothole segmentation model for **real-time road damage assessment**, particularly focusing on **potholes**. Our final goal will be to continuously calculate the area and percentage of pothole damage in each video frame. This accurate identification and quantification will support road repair strategies and provide immediate hazard warnings, enhancing road safety and aiding smart city development.

Initially, I'll demonstrate our approach using a sample image, applying our best model to detect, and display potholes with segmented masks representing the damaged areas:

In [ ]:
```
# Define the path to the validation images
valid_images_path = os.path.join(dataset_path, 'valid', 'images')

# List all jpg images in the directory
image_files = [file for file in os.listdir(valid_images_path) if file.endswith('.jpg')]

# Select a sample image
selected_image = image_files[45]

# Perform inference on the selected image
image_path = os.path.join(valid_images_path, selected_image)
results = best_model.predict(source=image_path, imgsz=640, conf=0.5)
annotated_image = results[0].plot()
annotated_image_rgb = cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB)

# Determine the number of subplots needed (1 original + number of masks)
num_subplots = 1 + (len(results[0].masks.data) if results[0].masks is not None else 0)

# Initialize the subplot with 1 row and n columns
fig, axes = plt.subplots(1, num_subplots, figsize=(15, 5))

# Display the original annotated image
axes[0].imshow(annotated_image_rgb)
axes[0].set_title('Original Image')
axes[0].axis('off')

# If multiple masks, iterate and display each mask
if results[0].masks is not None:
    masks = results[0].masks.data.cpu().numpy()
    for i, mask in enumerate(masks):
        # Threshold the mask to make sure it's binary
        # Any value greater than 0 is set to 255, else it remains 0
        binary_mask = (mask > 0).astype(np.uint8) * 255
        axes[i+1].imshow(binary_mask, cmap='gray')
        axes[i+1].set_title(f'Segmented Mask {i+1}')
        axes[i+1].axis('off')

# Adjust layout and display the subplot
plt.tight_layout()
plt.show()
```

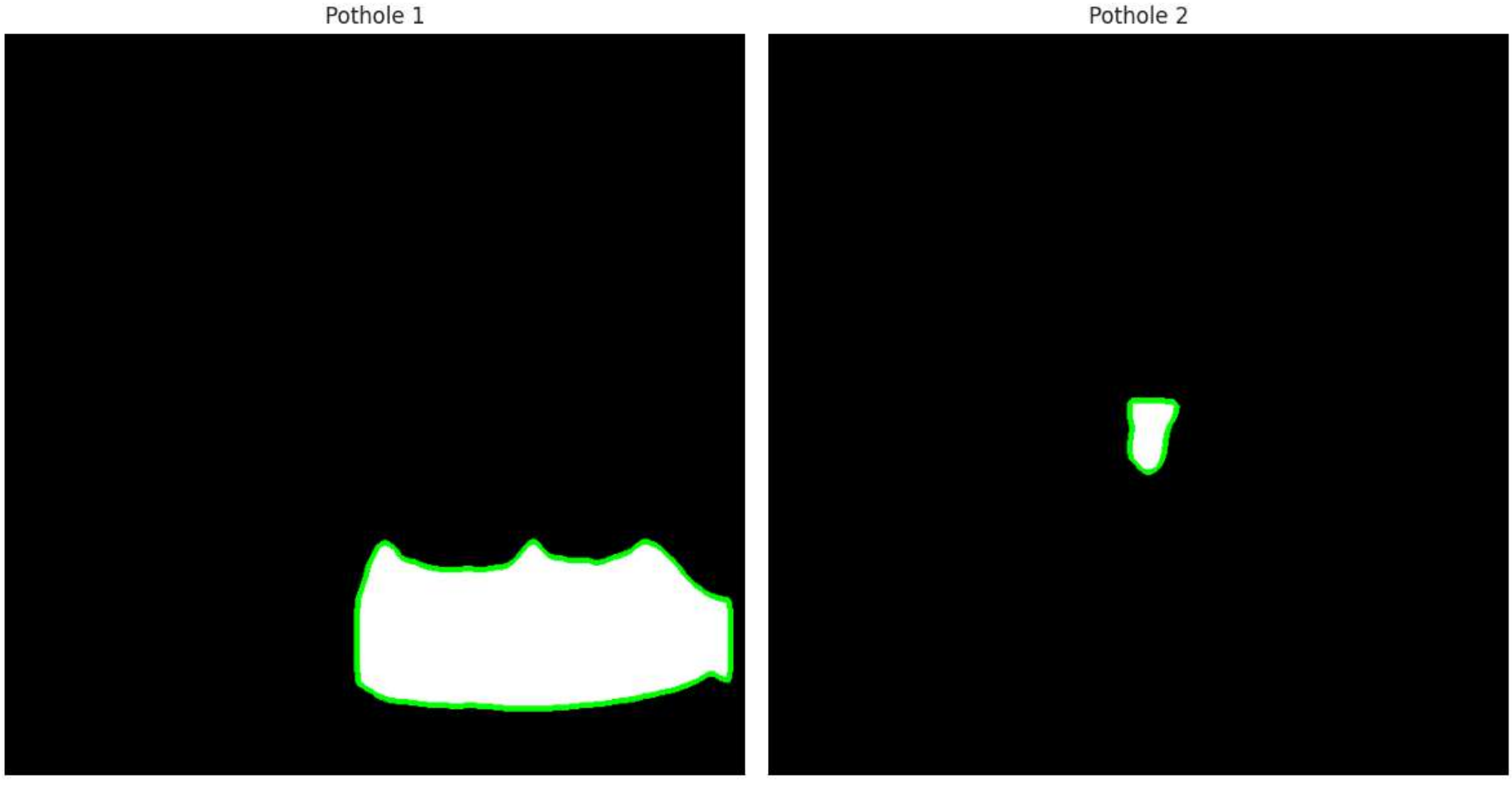
```
image 1/1 /kaggle/input/pothole-image-segmentation-dataset/Pothole_Segmentation_YOLOv8/valid/images/pic-50-_jpg.rf.cb41506e84f6
cac629bd55e40e329834.jpg: 640x640 2 Potholes, 8.6ms
Speed: 1.7ms preprocess, 8.6ms inference, 2.1ms postprocess per image at shape (1, 3, 640, 640)
```



Next, I will outline each detected pothole with contours on the masks, calculate the area of each, and ultimately determine both the total damaged area and the percentage of road damage due to potholes:

In [ ]:

```python
# Initialize variables to hold total area and individual areas
total_area = 0
area_list = []

# Set up the subplot for displaying masks
fig, axes = plt.subplots(1, len(masks), figsize=(12, 8))

# Perform operations if masks are available
if results[0].masks is not None:
    masks = results[0].masks.data.cpu().numpy()   # Retrieve masks as numpy arrays
    image_area = masks.shape[1] * masks.shape[2]  # Calculate total number of pixels in the image
    for i, mask in enumerate(masks):
        binary_mask = (mask > 0).astype(np.uint8) * 255  # Convert mask to binary
        color_mask = cv2.cvtColor(binary_mask, cv2.COLOR_GRAY2BGR)  # Convert binary mask to color
        contours, _ = cv2.findContours(binary_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)  # Find contours in the binary mas
        contour = contours[0]  # Retrieve the first contour
        area = cv2.contourArea(contour)  # Calculate the area of the pothole
        area_list.append(area)  # Append area to the list
        cv2.drawContours(color_mask, [contour], -1, (0, 255, 0), 3)  # Draw the contour on the mask

        # Display the mask with the green contour
        axes[i].imshow(color_mask)
        axes[i].set_title(f'Pothole {i+1}')
        axes[i].axis('off')

# Display all masks
plt.tight_layout()
plt.show()

# Calculate and print areas after displaying the images
for i, area in enumerate(area_list):
    print(f"Area of Pothole {i+1}: {area} pixels")
    total_area += area  # Sum the areas for total

# Calculate and print the total damaged area and percentage of road damaged by potholes
print("-"*50)
print(f"Total Damaged Area by Potholes: {total_area} pixels")
print(f"Total Pixels in Image: {image_area} pixels")
print(f"Percentage of Road Damaged: {(total_area / image_area) * 100:.2f}%")
```




```
Area of Pothole 1: 37883.5 pixels
Area of Pothole 2: 1894.5 pixels
-------------------------------------------------
Total Damaged Area by Potholes: 39778.0 pixels
Total Pixels in Image: 409600 pixels
Percentage of Road Damaged: 9.71%
```

So far, we've demonstrated how to apply our segmentation model to each image, extract masks for potholes, outline them with contours, and calculate each's area. By summing these areas, we estimate the road damage percentage caused by potholes. Next, I'll extend this approach to each frame of an unseen video, enabling real-time road damage assessment:

In [ ]:

```python
# Define the video path
video_path = '/kaggle/working/sample_video.mp4'

# Define font, scale, colors, and position for the annotation
font = cv2.FONT_HERSHEY_SIMPLEX
font_scale = 1
text_position = (40, 80)
font_color = (255, 255, 255)    # White color for text
background_color = (0, 0, 255)  # Red background for text

# Initialize a deque with fixed length for averaging the last 10 percentage damages
damage_deque = deque(maxlen=10)

# Open the video
cap = cv2.VideoCapture(video_path)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('road_damage_assessment.avi', fourcc, 20.0, (int(cap.get(3)), int(cap.get(4))))

# Read until video is completed
while cap.isOpened():
     # Capture frame-by-frame
    ret, frame = cap.read()
    if ret:
        # Perform inference on the frame
        results = best_model.predict(source=frame, imgsz=640, conf=0.25)
        processed_frame = results[0].plot(boxes=False)

        # Initializes percentage_damage to 0
        percentage_damage = 0

        # If masks are available, calculate total damage area and percentage
        if results[0].masks is not None:
            total_area = 0
            masks = results[0].masks.data.cpu().numpy()
            image_area = frame.shape[0] * frame.shape[1]  # total number of pixels in the image
            for mask in masks:
                binary_mask = (mask > 0).astype(np.uint8) * 255
                contour, _ = cv2.findContours(binary_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

```
                total_area += cv2.contourArea(contour[0])

            percentage_damage = (total_area / image_area) * 100

        # Calculate and update the percentage damage
        damage_deque.append(percentage_damage)
        smoothed_percentage_damage = sum(damage_deque) / len(damage_deque)

        # Draw a thick line for text background
        cv2.line(processed_frame, (text_position[0], text_position[1] - 10),
                 (text_position[0] + 350, text_position[1] - 10), background_color, 40)

        # Annotate the frame with the percentage of damage
        cv2.putText(processed_frame, f'Road Damage: {smoothed_percentage_damage:.2f}%', text_position, font, font_scale, font_

        # Write the processed frame to the output video
        out.write(processed_frame)

        # Uncomment the following 3 lines if running this code on a local machine to view the real-time processing results
        # cv2.imshow('Road Damage Assessment', processed_frame) # Display the processed frame
        # if cv2.waitKey(1) & 0xFF == ord('q'): # Press Q on keyboard to exit the loop
        #     break
    else:
        break

# Release the video capture and video write objects
cap.release()
out.release()

# Close all the frames
# cv2.destroyAllWindows()
```

```
0: 384x640 1 Pothole, 12.7ms
Speed: 2.4ms preprocess, 12.7ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.6ms
Speed: 1.9ms preprocess, 11.6ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.8ms
Speed: 1.8ms preprocess, 12.8ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 10.8ms
Speed: 1.4ms preprocess, 10.8ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 10.9ms
Speed: 1.4ms preprocess, 10.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.6ms preprocess, 11.2ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.7ms
Speed: 1.8ms preprocess, 11.7ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 13.2ms
Speed: 1.8ms preprocess, 13.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 10.9ms
Speed: 1.4ms preprocess, 10.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.8ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.9ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.7ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 10.9ms
Speed: 1.8ms preprocess, 10.9ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.9ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.9ms
Speed: 1.8ms preprocess, 11.9ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.5ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.6ms
Speed: 2.2ms preprocess, 12.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 4 Potholes, 11.3ms
Speed: 1.6ms preprocess, 11.3ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.7ms
Speed: 1.8ms preprocess, 12.7ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 2.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.4ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.2ms
Speed: 1.8ms preprocess, 12.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.6ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 13.0ms
Speed: 1.7ms preprocess, 13.0ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.7ms
Speed: 1.8ms preprocess, 11.7ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.6ms
Speed: 1.7ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 5 Potholes, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 12.9ms
Speed: 1.9ms preprocess, 12.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 12.9ms
Speed: 1.8ms preprocess, 12.9ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.5ms
Speed: 1.8ms preprocess, 12.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.7ms
Speed: 1.7ms preprocess, 12.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 2.2ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 10.9ms
Speed: 1.5ms preprocess, 10.9ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.8ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.9ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.4ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 2 Potholes, 12.5ms
Speed: 1.7ms preprocess, 12.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.5ms preprocess, 11.3ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.6ms preprocess, 11.3ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.4ms
Speed: 1.4ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.4ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.1ms
Speed: 1.7ms preprocess, 12.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.1ms
Speed: 1.7ms preprocess, 12.1ms inference, 3.1ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 14.5ms
Speed: 1.9ms preprocess, 14.5ms inference, 3.3ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 2.1ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.5ms
Speed: 1.7ms preprocess, 12.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 2 Potholes, 13.0ms
Speed: 1.8ms preprocess, 13.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.7ms
Speed: 1.9ms preprocess, 12.7ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 2.0ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.7ms
Speed: 1.8ms preprocess, 12.7ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.9ms
Speed: 1.8ms preprocess, 12.9ms inference, 0.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.8ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.7ms
Speed: 1.8ms preprocess, 11.7ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.0ms
Speed: 1.6ms preprocess, 11.0ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.6ms
Speed: 1.5ms preprocess, 11.6ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 10.9ms
Speed: 1.5ms preprocess, 10.9ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 10.9ms
Speed: 1.5ms preprocess, 10.9ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 10.9ms
Speed: 1.5ms preprocess, 10.9ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.6ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.2ms
Speed: 1.4ms preprocess, 11.2ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.5ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 13.1ms
Speed: 1.9ms preprocess, 13.1ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 13.4ms
Speed: 1.9ms preprocess, 13.4ms inference, 2.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 14.0ms
Speed: 1.9ms preprocess, 14.0ms inference, 3.1ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 2 Potholes, 13.6ms
Speed: 1.9ms preprocess, 13.6ms inference, 2.3ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 13.8ms
Speed: 1.9ms preprocess, 13.8ms inference, 2.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 13.0ms
Speed: 1.7ms preprocess, 13.0ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.6ms
Speed: 1.8ms preprocess, 12.6ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 13.4ms
Speed: 1.9ms preprocess, 13.4ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.6ms
Speed: 1.9ms preprocess, 12.6ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.9ms
Speed: 1.8ms preprocess, 11.9ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.6ms
Speed: 1.8ms preprocess, 12.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 13.2ms
Speed: 2.6ms preprocess, 13.2ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.1ms
Speed: 1.7ms preprocess, 12.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.0ms
Speed: 1.7ms preprocess, 12.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.6ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.8ms
Speed: 1.7ms preprocess, 12.8ms inference, 2.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.9ms
Speed: 1.6ms preprocess, 12.9ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.9ms
Speed: 2.0ms preprocess, 12.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.8ms
Speed: 1.5ms preprocess, 11.8ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.4ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.7ms
Speed: 1.6ms preprocess, 12.7ms inference, 2.1ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 13.0ms
Speed: 1.9ms preprocess, 13.0ms inference, 0.7ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 (no detections), 11.6ms
Speed: 1.9ms preprocess, 11.6ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.9ms
Speed: 1.9ms preprocess, 12.9ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.6ms
Speed: 1.8ms preprocess, 12.6ms inference, 0.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.8ms
Speed: 1.7ms preprocess, 12.8ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.2ms
Speed: 1.8ms preprocess, 12.2ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.6ms
Speed: 1.7ms preprocess, 12.6ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.1ms
Speed: 1.8ms preprocess, 12.1ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.2ms
Speed: 1.5ms preprocess, 11.2ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.8ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.6ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.0ms
Speed: 1.8ms preprocess, 12.0ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.8ms
Speed: 1.7ms preprocess, 12.8ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 10.9ms
Speed: 1.5ms preprocess, 10.9ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.8ms
Speed: 1.8ms preprocess, 12.8ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.7ms
Speed: 1.8ms preprocess, 11.7ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 1 Pothole, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.4ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.6ms
Speed: 1.7ms preprocess, 12.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.9ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.6ms
Speed: 1.7ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.4ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.0ms
Speed: 1.6ms preprocess, 11.0ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 13.2ms
Speed: 1.7ms preprocess, 13.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.6ms
Speed: 1.5ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.6ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.7ms
Speed: 1.8ms preprocess, 11.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 13.0ms
Speed: 1.8ms preprocess, 13.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.4ms preprocess, 11.0ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.8ms
Speed: 1.7ms preprocess, 12.8ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 3 Potholes, 11.7ms
Speed: 1.5ms preprocess, 11.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 10.9ms
Speed: 1.4ms preprocess, 10.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 2.0ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 2.0ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.5ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 2.0ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 12.6ms
Speed: 1.8ms preprocess, 12.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.9ms
Speed: 1.9ms preprocess, 11.9ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.9ms preprocess, 11.3ms inference, 2.3ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.3ms
Speed: 1.5ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 12.8ms
Speed: 1.6ms preprocess, 12.8ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.3ms
Speed: 1.6ms preprocess, 11.3ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 12.6ms
Speed: 1.8ms preprocess, 12.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 3 Potholes, 10.9ms
Speed: 1.4ms preprocess, 10.9ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.6ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 12.6ms
Speed: 1.7ms preprocess, 12.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.5ms preprocess, 11.2ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 2.0ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.7ms
Speed: 1.8ms preprocess, 12.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.3ms
Speed: 1.7ms preprocess, 12.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 2.1ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.9ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.7ms
Speed: 1.5ms preprocess, 11.7ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 10.9ms
Speed: 1.7ms preprocess, 10.9ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 1 Pothole, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.3ms
Speed: 1.5ms preprocess, 11.3ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 10.9ms
Speed: 1.5ms preprocess, 10.9ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.2ms
Speed: 1.5ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 0.5ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.9ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.7ms
Speed: 2.0ms preprocess, 11.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.7ms
Speed: 1.8ms preprocess, 12.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.1ms
Speed: 1.8ms preprocess, 12.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.4ms
Speed: 1.7ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.6ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 12.4ms
Speed: 1.7ms preprocess, 12.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.5ms
Speed: 1.9ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.2ms
Speed: 1.8ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.6ms
Speed: 1.9ms preprocess, 11.6ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 13.0ms
Speed: 1.8ms preprocess, 13.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.9ms
Speed: 1.5ms preprocess, 12.9ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 6 Potholes, 11.1ms
Speed: 2.1ms preprocess, 11.1ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 12.9ms
Speed: 1.8ms preprocess, 12.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.7ms
Speed: 1.8ms preprocess, 11.7ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.2ms
Speed: 1.5ms preprocess, 11.2ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 3 Potholes, 11.2ms
Speed: 1.6ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.7ms
Speed: 1.8ms preprocess, 12.7ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.4ms
Speed: 1.7ms preprocess, 12.4ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.7ms
Speed: 1.7ms preprocess, 11.7ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.3ms
Speed: 1.4ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.8ms preprocess, 11.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 12.9ms
Speed: 1.8ms preprocess, 12.9ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 13.3ms
Speed: 1.7ms preprocess, 13.3ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.8ms
Speed: 1.8ms preprocess, 11.8ms inference, 2.2ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.8ms
Speed: 1.8ms preprocess, 11.8ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 12.5ms
Speed: 1.8ms preprocess, 12.5ms inference, 1.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.7ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.0ms
Speed: 1.9ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.2ms
Speed: 2.2ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.8ms preprocess, 11.6ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.9ms
Speed: 1.8ms preprocess, 11.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.2ms
Speed: 1.9ms preprocess, 11.2ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.1ms
Speed: 1.7ms preprocess, 11.1ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.7ms
Speed: 1.7ms preprocess, 11.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
```

```
0: 384x640 1 Pothole, 11.2ms
Speed: 1.7ms preprocess, 11.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.7ms
Speed: 1.7ms preprocess, 11.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.8ms
Speed: 1.6ms preprocess, 11.8ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.8ms preprocess, 11.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.7ms preprocess, 11.0ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.7ms
Speed: 1.7ms preprocess, 12.7ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.5ms
Speed: 1.7ms preprocess, 11.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.9ms
Speed: 1.8ms preprocess, 11.9ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 Potholes, 11.6ms
Speed: 1.7ms preprocess, 11.6ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.8ms
Speed: 1.8ms preprocess, 12.8ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 12.5ms
Speed: 1.8ms preprocess, 12.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.1ms
Speed: 1.5ms preprocess, 11.1ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.0ms
Speed: 1.5ms preprocess, 11.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.4ms
Speed: 1.8ms preprocess, 11.4ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 11.3ms
Speed: 1.7ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.2ms
Speed: 1.6ms preprocess, 11.2ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 12.2ms
Speed: 1.7ms preprocess, 12.2ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 10.8ms
Speed: 1.4ms preprocess, 10.8ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 Pothole, 10.8ms
Speed: 1.4ms preprocess, 10.8ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 Potholes, 11.3ms
Speed: 1.5ms preprocess, 11.3ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 (no detections), 11.1ms
Speed: 1.6ms preprocess, 11.1ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)
```

Finally, lets convert the output `.avi` video to `.mp4` format for notebook playback and display the result:

In [ ]:
```
# Convert the .avi video generated by our traffic density estimation app to .mp4 format for compatibility with notebook displa
!ffmpeg -y -loglevel panic -i /kaggle/working/road_damage_assessment.avi road_damage_assessment.mp4

# Embed and display the processed sample video within the notebook
Video("road_damage_assessment.mp4", embed=True, width=960)
```

Out[ ]: