YOLO-NAS with DeepSORT Tracking

In this project, we will implement

1. Object Detection with YOLO-NAS
2. Object Tracking with YOLO-NAS and DeepSORT
3. Vehicles Counting (Entering and Leaving) with YOLO-NAS and DeepSORT
4. Vehicles Counting (Entering and Leaving) with YOLO-NAS and SORT
5. YOLO-NAS for Object Tracking on a Custom Dataset (Ships Detection)

Install All the Required Packages

The code below will install two packages which include SuperGradients and filterpy package. To do object detection with the YOLO-NAS on images and videos, SuperGradients package is used and to implement object tracking with SORT algorithm, a filterpy package is required. Filterpy package is used for implementing and testing Kalman filter

```
!pip install -q super-gradients==3.2.1
!pip install filterpy==1.1.0
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.4/6.4 MB 48.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.3/139.3 kB 13.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 684.5/684.5 kB 23.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 70.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 25.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 408.6/408.6 kB 32.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.5/154.5 kB 17.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.5/79.5 kB 9.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 36.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.5/13.5 MB 25.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 68.0/68.0 kB 7.3 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.0/17.0 MB 69.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 68.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 99.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 5.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.8/11.8 MB 102.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82.0/82.0 kB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.0/117.0 kB 15.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 575.5/575.5 kB 60.3 MB/s eta 0:00:00
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 103.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.1/121.1 kB 15.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 10.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.0/120.0 kB 16.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.0/120.0 kB 15.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.6/120.6 kB 14.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.5/83.5 kB 8.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.5/83.5 kB 9.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.7/84.7 kB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 kB 13.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.2/99.2 kB 10.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.8/99.8 kB 11.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.4/89.4 kB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.4/89.4 kB 11.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 90.6/90.6 kB 11.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 kB 11.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 kB 10.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.6/92.6 kB 10.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.0/94.0 kB 12.3 MB/s eta 0:00:00
  Building wheel for pycocotools (pyproject.toml) ... done
  Building wheel for termcolor (setup.py) ... done
  Building wheel for treelib (setup.py) ... done
  Building wheel for coverage (setup.py) ... done
  Building wheel for antlr4-python3-runtime (setup.py) ... done
  Building wheel for stringcase (setup.py) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages tha
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires kaleido, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
```

Import All the Required Libraries

In the code below, we will import all the required libraries which includes opencv, super gradients and numpy libraries. To read or display an input or output image or to do image processing we require the opencv-python package. Along with this to read an input video, frame by frame or to display or save the output video opencv-python package is also used. As YOLO-NAS is available under the super gradients package which is an open source computer vision training library, we will import models from super gradients training library that will enable us to access YOLO-NAS model pretrained weights.

```
import cv2
import torch
from super_gradients.training import models
import numpy as np
import math
from numpy import random
from IPython.display import HTML
from base64 import b64encode
import os
```

```
[2023-12-01 03:11:46] INFO - crash_tips_setup.py - Crash tips is enabled. You can set your e
The console stream is logged into /root/sg_logs/console.log
[2023-12-01 03:11:46] WARNING - __init__.py - Failed to import pytorch_quantization
[2023-12-01 03:11:46] INFO - utils.py - NumExpr defaulting to 2 threads.
```

```
/usr/local/lib/python3.10/dist-packages/_distutils_hack/__init__.py:33: UserWarning: Setupto
  warnings.warn("Setuptools is replacing distutils.")
[2023-12-01 03:11:54] WARNING - calibrator.py - Failed to import pytorch_quantization
[2023-12-01 03:11:54] WARNING - export.py - Failed to import pytorch_quantization
[2023-12-01 03:11:54] WARNING - selective_quantization_utils.py - Failed to import pytorch_q
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: h
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: c
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: c
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: s
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: t
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: h
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: c
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: c
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: c
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: e
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: t
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: s
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: n
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: j
[2023-12-01 03:11:54] WARNING - env_sanity_check.py - Failed to verify installed packages: c
```

Download the Demo Videos

The code below will download sample videos from the drive in the notebook and we will be using these to do Object Detection, Vehicles Counting and Ship Detection

```
!gdown "https://drive.google.com/uc?id=1Uxf5h5hc7DoZ9ken84L2JGzNNE4O0Z6j&confirm=t"
!gdown "https://drive.google.com/uc?id=1RsY2ZhwKKgWIBLi7wFC8bdgDvcofT9fd&confirm=t"
!gdown "https://drive.google.com/uc?id=1hkYL8qQdZ1UCPvg-6vtFh811jwSJhMi0&confirm=t"
!gdown "https://drive.google.com/uc?id=1xFQiefMgbYNLyReYKs456P5Dy_snnJn-&confirm=t"
```

```
Downloading...
From: https://drive.google.com/uc?id=1Uxf5h5hc7DoZ9ken84L2JGzNNE4O0Z6j&confirm=t
To: /content/bikes.mp4
100% 3.14M/3.14M [00:00<00:00, 183MB/s]
Downloading...
From: https://drive.google.com/uc?id=1RsY2ZhwKKgWIBLi7wFC8bdgDvcofT9fd&confirm=t
To: /content/NewVehiclesEnteringandLeaving.mp4
100% 11.0M/11.0M [00:00<00:00, 112MB/s]
Downloading...
From: https://drive.google.com/uc?id=1hkYL8qQdZ1UCPvg-6vtFh811jwSJhMi0&confirm=t
To: /content/video1.mp4
100% 8.62M/8.62M [00:00<00:00, 78.4MB/s]
Downloading...
From: https://drive.google.com/uc?id=1xFQiefMgbYNLyReYKs456P5Dy_snnJn-&confirm=t
To: /content/test3.mp4
100% 2.07M/2.07M [00:00<00:00, 188MB/s]
```

Load the Input Video and calculate the Frame Width and Frame Height

To load the input video and calculate the frame width and frame height. We will create a function which the video path as the input and returns the video, frame width and frame height

```
def get_video_info(video_path):
    # Open the video file
    cap = cv2.VideoCapture(video_path)

    # Check if the video file is opened successfully
    if not cap.isOpened():
        raise ValueError("Error: Could not open the video file.")

    # Get the frame width and frame height
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    return cap, frame_width, frame_height
```

```
video_path = '/content/output_video_combined (2).mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

Download the Pre-Trained YOLO-NAS Model Weights

Since we have installed all the required packages and imported all the required libraries, we will download the pre-trained YOLO-NAS model weights. YOLO-NAS comes with three different models, YOLO-NAS (small, medium, large). YOLO-NAS-S is the fastest but it is less accurate as compared to other YOLO-NAS models. YOLO-NAS-L is the most accurate among other YOLO-NAS models but it is less fast as compared to other YOLO-NAS models. We will use the small and large version of the YOLO-NAS models and the pre-trained weights from the COCO dataset in this project.

```
def load_model(model_name):
    # Check if CUDA (GPU) is available and set the device accordingly
    device = torch.device("cuda:0") if torch.cuda.is_available() else torch.device("cpu")

    # Load the specified model
    model = models.get(model_name, pretrained_weights="coco").to(device)
    return model
```

```
model_name = 'yolo_nas_s'
model = load_model(model_name)
```

```
[2023-12-01 03:14:41] INFO - checkpoint_utils.py - License Notification: YOLO-NAS pre-traine
https://github.com/Deci-AI/super-gradients/blob/master/LICENSE.YOLONAS.md
By downloading the pre-trained weight files you agree to comply with these terms.
Downloading: "https://sghub.deci.ai/models/yolo_nas_s_coco.pth" to /root/.cache/torch/hub/ch
100%|██████████| 73.1M/73.1M [00:00<00:00, 126MB/s]
[2023-12-01 03:14:42] INFO - checkpoint_utils.py - Successfully loaded pretrained weights fo
```

1. Object Detection with YOLO-NAS

1.A. Classes in the COCO Dataset

```
def cococlassNames():
  class_names = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "tru
  return class_names
```

```
classNames = cococlassNames()
```

1.B. Prediction

To perform object detection on a video, will loop through each of the frames one by one and do the detection of the objects in each of the frames and using opencv write function, we will also save the output video as well.

```
out=cv2.VideoWriter('output.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 20, (frame_w:
count=0
while True:
  ret, frame=cap.read()
  count+=1
  if ret:
    result=list(model.predict(frame, conf=0.5))[0]
    bbox_xyxys=result.prediction.bboxes_xyxy.tolist()
    confidences=result.prediction.confidence
    labels=result.prediction.labels.tolist()
    for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
      bbox=np.array(bbox_xyxy)
      x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
      x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
      classname=int(cls)
      class_name=classNames[classname]
      conf=math.ceil((confidence*100))/100
      label=f'{class_name}{conf}'
      print("Frame N", count, "", x1, y1,x2, y2)
      #Create Bounding Boxes around the Detected Objects
      cv2.rectangle(frame, (x1, y1), (x2, y2), color=(0, 0, 255, 0.6),thickness=2, lineType
      #Create a rectangle above the detected object and add label and confidence score
      t_size=cv2.getTextSize(str(label), 0, fontScale=1/2, thickness=1)[0]
      c2=x1+t_size[0], y1-t_size[1]-3
      cv2.rectangle(frame, (x1, y1), c2, color=(0, 0, 255, 0.6), thickness=-1, lineType=cv2
      cv2.putText(frame, str(label), (x1, y1-2), 0, 1/2, [255, 255, 255], thickness=1, line
    out.write(frame)
  else:
    break
out.release()
cap.release()
```

```
[2023-12-01 03:14:59] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-12-01 03:15:06] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 1  226 46 245 103
Frame N 1  256 38 274 98
Frame N 2  226 46 245 103
Frame N 2  257 38 274 98
[2023-12-01 03:15:06] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-12-01 03:15:06] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 3  226 46 245 103
Frame N 3  257 38 274 98
Frame N 4  226 46 245 102
Frame N 4  257 40 274 98
[2023-12-01 03:15:06] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-12-01 03:15:06] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 5  226 46 246 102
Frame N 5  257 47 274 97
Frame N 6  226 46 246 102
Frame N 6 [2023-12-01 03:15:07] INFO - pipelines.py - Fusing some of the model's layers. I
 257 46 274 97
[2023-12-01 03:15:07] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 7  226 45 246 102
Frame N 7  255 57 274 97
Frame N 8  226 45 246 102
Frame N 8  256 57 274 96
[2023-12-01 03:15:07] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-12-01 03:15:07] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 9  226 45 246 102
Frame N 9  256 56 274 97
[2023-12-01 03:15:08] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 10  226 45 246 102
Frame N 10  256 56 274 97
[2023-12-01 03:15:08] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 11  226 44 246 102
[2023-12-01 03:15:08] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 12  226 44 246 101
Frame N 12  0 128 24 167
[2023-12-01 03:15:09] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 13  226 44 246 101
Frame N 13  0 126 31 169
Frame N 13  443 0 502 44
Frame N 13  442 0 502 44
[2023-12-01 03:15:09] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 14  227 44 245 101
Frame N 14  0 123 38 170
[2023-12-01 03:15:11] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 15  227 44 245 101
Frame N 15  0 124 46 171
[2023-12-01 03:15:12] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 16  226 44 245 101
Frame N 16  0 124 46 171
[2023-12-01 03:15:13] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 17  227 43 246 101
Frame N 17  0 121 54 173
[2023-12-01 03:15:13] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Frame N 18  227 43 246 101
Frame N 18  0 120 62 174
Frame N 18  429 4 490 51
```

1.C. Display the Output Demo Video

```
!rm '/content/result_compressed.mp4'
```

```
rm: cannot remove '/content/result_compressed.mp4': No such file or directory
```

```
def display_compressed_video(input_path):
    # Compressed video path
    compressed_path = "/content/output.avi"

    os.system(f"ffmpeg -i {input_path} -vcodec libx264 {compressed_path}")

    # Show video
    mp4 = open(compressed_path, 'rb').read()
    data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
    display_html = """
    <video width=400 controls>
        <source src="%s" type="video/mp4">
    </video>
    """ % data_url

    return HTML(display_html)
```

```
input_video_path = '/content/output.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```

2.YOLO-NAS with DeepSORT Tracking

DeepSORT is a Computer Vision Tracking Algorithm used to track the objects while assigning each of the tracked objects a unique id. DeepSORT is an Multiple Object Tracking (MOT) algorithm which is an extension of the SORT algorithm. DeepSORT introduces deep learning into the SORT algorithm by adding an appearance descriptor to reduce the identity switches and hence making the tracking more efficient. DeepSORT uses a better association metric that combines both motion and appearance descriptors. DeepSORT can be defined as the tracking algorithm which tracks objects not only based on the velocity and motion of the object but also the appearance of the object.

2.A. To implement the Object Tracking with DeepSORT, Download the DeepSORT files

In object tracking a unique id is assigned to each of the detected objects. To implement object tracking, we first get an initial set of detections then we assign a unique id to each of the detected

objects and track the detected objects throughout the frames of the video feed while maintaining the assigned ids.

```
!gdown "https://drive.google.com/uc?id=11ZSZcG-bcbueXZC3rN08CM0qqX3eiHxf&confirm=t"
```

```
Downloading...
From: https://drive.google.com/uc?id=11ZSZcG-bcbueXZC3rN08CM0qqX3eiHxf&confirm=t
To: /content/deep_sort_pytorch.zip
 40% 17.3M/43.1M [00:00<00:01, 20.4MB/s]
```

```
!unzip /content/deep_sort_pytorch.zip
```

2.B. Import all the DeepSORT Required Libraries

get_config library contains functions or classes for parsing configuration files and command-line arguments related to Deep SORT tracking

deep_sort_pytorch.deep_sort contains contains classes and functions for initializing the tracker, processing detections, associating detections with existing tracks, updating track states, and producing tracking results.

```
from deep_sort_pytorch.utils.parser import get_config
from deep_sort_pytorch.deep_sort import DeepSort
```

2.C. Load the Input Video and Calculate the Frame Width and Frame Height

```
video_path = '/content/test3.mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

2.D. Download the Pre-Trained YOLO-NAS Model Weights

```
model_name = 'yolo_nas_l'
model = load_model(model_name)
```

```
[2023-09-27 06:46:09] INFO - checkpoint_utils.py - License Notification: YOLO-NAS pre-traine
https://github.com/Deci-AI/super-gradients/blob/master/LICENSE.YOLONAS.md
By downloading the pre-trained weight files you agree to comply with these terms.
Downloading: "https://sghub.deci.ai/models/yolo_nas_l_coco.pth" to /root/.cache/torch/hub/ch
100%|[redacted]| 256M/256M [00:09<00:00, 29.7MB/s]
[2023-09-27 06:46:19] INFO - checkpoint_utils.py - Successfully loaded pretrained weights fo
```

2.E. Initialize the DeepSORT

Configuring and Initializing the instance of the Deep SORT tracker

The code below sets up and initializes a Deep SORT tracker with various configuration options and parameters, allowing it to track objects in a video sequence with the specified settings.

```
def initialize_deepsort():
    # Create the Deep SORT configuration object and load settings from the YAML file
    cfg_deep = get_config()
    cfg_deep.merge_from_file("/content/deep_sort_pytorch/configs/deep_sort.yaml")

    # Initialize the DeepSort tracker
    deepsort = DeepSort(cfg_deep.DEEPSORT.REID_CKPT,
                        max_dist=cfg_deep.DEEPSORT.MAX_DIST,
                        # min_confidence  parameter sets the minimum tracking confidence re
                        min_confidence=cfg_deep.DEEPSORT.MIN_CONFIDENCE,
                        #nms_max_overlap specifies the maximum allowed overlap between bour
                        nms_max_overlap=cfg_deep.DEEPSORT.NMS_MAX_OVERLAP,
                        #max_iou_distance parameter defines the maximum intersection-over-u
                        max_iou_distance=cfg_deep.DEEPSORT.MAX_IOU_DISTANCE,
                        # Max_age: If an object's tracking ID is lost (i.e., the object is
                        max_age=cfg_deep.DEEPSORT.MAX_AGE, n_init=cfg_deep.DEEPSORT.N_INIT,
                        #nn_budget: It sets the budget for the nearest-neighbor search.
                        nn_budget=cfg_deep.DEEPSORT.NN_BUDGET,
                        use_cuda=True
            )

    return deepsort
```

```
deepsort = initialize_deepsort()
```

```
[2023-09-27 06:48:05] INFO - feature_extractor.py - Loading weights from deep_sort_pytorch/c
```

2.F. Creating Helping Functions

A helper function draw boxes is created, to draw bounding around each of the detected objects. After getting an initial set of detections with YOLO-NAS, we assign a unique id to each of the detected objects. Along with this we will create a function compute_color_for_labels to create a fixed color bounding box depending on the class

```
names = cococlassNames()
colors = [[random.randint(0, 255) for _ in range(3)]
          for _ in range(len(names))]
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
```

```
def compute_color_for_labels(label):
    """
    Function that adds fixed color depending on the class
    """
    if label == 0:  # person  #BGR
        color = (85, 45, 255)
    elif label == 2:  # Car
        color = (222, 82, 175)
    elif label == 3:  # Motobike
        color = (0, 204, 255)
    elif label == 5:  # Bus
        color = (0, 149, 255)
    else:
        color = [int((p * (label ** 2 - label + 1)) % 255) for p in palette]
    return tuple(color)
```

```
#Creating a helper function

def draw_boxes(img, bbox, identities=None, categories=None, names=None, offset=(0,0)):
    for i, box in enumerate(bbox):
        x1, y1,  x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[0]
        y2 += offset[0]
        cat = int(categories[i]) if categories is not None else 0
        id = int(identities[i]) if identities is not None else 0
        #Create Bounding Boxes around the Detected Objects
        cv2.rectangle(img, (x1, y1), (x2, y2), color= compute_color_for_labels(cat),thickne
        label = str(id) + ":" + classNames[cat]
        (w,h), _ = cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, th
        #Create a rectangle above the detected object and add label and confidence score
        t_size=cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, thickne
        c2=x1+t_size[0], y1-t_size[1]-3
        cv2.rectangle(frame, (x1, y1), c2, color=compute_color_for_labels(cat), thickness=
        cv2.putText(frame, str(label), (x1, y1-2), 0, 1/2, [255, 255, 255], thickness=1, l
    return img
```

2.G. Object Tracking with YOLO-NAS and DeepSORT

In the code below, we will first upload a video and get an initial set of detections with YOLO-NAS then we assign a unique id to each of the detected objects and track the detected objects throughout the frames of the video feed while maintaining the assigned ids.

```
classNames = cococlassNames()
output = cv2.VideoWriter('Output1.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (fr

while True:
    xywh_bboxs = []
    confs = []
    oids = []
    outputs = []
```

```
        ret, frame = cap.read()
        if ret:
          result = list(model.predict(frame, conf=0.5))[0]
          bbox_xyxys = result.prediction.bboxes_xyxy.tolist()
          confidences = result.prediction.confidence
          labels = result.prediction.labels.tolist()
          for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
              bbox = np.array(bbox_xyxy)
              x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
              x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
              conf = math.ceil((confidence*100))/100
              cx, cy = int((x1+x2)/2), int((y1+y2)/2)
              bbox_width = abs(x1-x2)
              bbox_height = abs(y1-y2)
              xcycwh = [cx, cy, bbox_width, bbox_height]
              xywh_bboxs.append(xcycwh)
              confs.append(conf)
              oids.append(int(cls))
          xywhs = torch.tensor(xywh_bboxs)
          confss= torch.tensor(confs)
          outputs = deepsort.update(xywhs, confss, oids, frame)
          if len(outputs)>0:
              bbox_xyxy = outputs[:,:4]
              identities = outputs[:, -2]
              object_id = outputs[:, -1]
              draw_boxes(frame, bbox_xyxy, identities, object_id)
          output.write(frame)
        else:
            break

    output.release()
    cap.release()
```

```
[2023-09-27 06:48:13] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:14] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:14] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:15] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:15] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:15] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:16] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:16] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:16] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:18] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:18] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:19] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:19] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:19] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:20] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:20] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:21] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:21] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:23] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:23] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

```
[2023-09-27 06:48:24] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:25] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:25] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:25] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:25] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:28] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:28] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:32] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:32] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:32] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:33] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:33] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:34] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:34] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:48:34] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

2.E. Display the Output Video

```
!rm '/content/result_compressed.mp4'
```

```
input_video_path = '/content/Output1.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```

3. Vehicles Counting Entering and Leaving with YOLO-NAS and DeepSORT Tracking

3.A. Load the input video and calculate the Frame Width and Frame Height

```
video_path = '/content/test3.mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

3.B. Download the Pre-Trained YOLO-NAS Model Weights and Initialize DeepSORT

```
model_name = 'yolo_nas_l'
model = load_model(model_name)
```

```
[2023-09-27 06:50:45] INFO - checkpoint_utils.py - License Notification: YOLO-NAS pre-traine
https://github.com/Deci-AI/super-gradients/blob/master/LICENSE.YOLONAS.md
By downloading the pre-trained weight files you agree to comply with these terms.
[2023-09-27 06:50:46] INFO - checkpoint_utils.py - Successfully loaded pretrained weights fo
```

```
deepsort = initialize_deepsort()
```

```
[2023-09-27 06:50:46] INFO - feature_extractor.py - Loading weights from deep_sort_pytorch/d
```

3.C. Creating Helper Functions

A helper function draw boxes is created, to draw bounding around each of the detected objects. After getting an initial set of detections with YOLO-NAS, we assign a unique id to each of the detected objects. And using these unique id's we will implement vehicles counting, we will save each of the unique id in a lsit and using the length of the list we will count the vehicles entering and leaving. Along with this we will create a function compute_color_for_labels to create a fixed color bounding box depending on the class

```
classNames = cococlassNames()
colors = [[random.randint(0, 255) for _ in range(3)]
          for _ in range(len(classNames))]
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
```

```
#Creating a helper function

def draw_boxes_vehicles_counting_el(img, bbox, identities=None, categories=None, names=None
    for i, box in enumerate(bbox):
        x1, y1,  x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[0]
        y2 += offset[0]
        cat = int(categories[i]) if categories is not None else 0
        id = int(identities[i]) if identities is not None else 0
        cx, cy = int((x1+x2)/2), int((y1+y2)/2)
        #cv2.circle(frame, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
        #Create Bounding Boxes around the Detected Objects
        cv2.rectangle(img, (x1, y1), (x2, y2), color= compute_color_for_labels(cat),thickne
        label = str(id) + ":" + classNames[cat]
        (w,h), _ = cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, th
        #Create a rectangle above the detected object and add label and confidence score
        t_size=cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, thickne
        c2=x1+t_size[0], y1-t_size[1]-3
        cv2.rectangle(img, (x1, y1), c2, color=compute_color_for_labels(cat), thickness=-1
        cv2.putText(img, str(label), (x1, y1-2), 0, 1/2, [255, 255, 255], thickness=1, line
        if limitup[0] < cx < limitup[2] and limitup[1] -4 < cy < limitup[3] + 4:
          if totalcountup.count(id)==0:
            totalcountup.append(id)
            cv2.line(img, (limitup[0], limitup[1]), (limitup[2], limitup[3]), (0,255, 0), 
        if limitdown[0] < cx < limitdown[2] and limitdown[1] -4 < cy < limitdown[3] + 4:
          if totalcountdown.count(id)==0:
            totalcountdown.append(id)
            cv2.line(img, (limitdown[0], limitdown[1]), (limitdown[2], limitdown[3]), (0,25
    cv2.rectangle(frame, (frame_width - 310,5), (frame_width,45), [0, 28, 136], -1, cv2.LIN
    cv2.putText(frame, str("Vehicle Entering") + ":" + str(len(totalcountup)), (frame_width
    cv2.rectangle(frame, (20,5), (310,45), [255, 0, 255], -1, cv2.LINE_AA)
    cv2.putText(frame, str("Vehicle Leaving") + ":" + str(len(totalcountdown)), (20, 35), (
    print("Total Count Up",len(totalcountup))
    print("Total Count Down",len(totalcountdown))
    return img
```

3.D. Test on Demo Video 1

The code snippet below will implement vechicles counting, we will create two dummy lines one to count the vehicles entering and one to count the vehicles leaving and when the vehicles intersect with any of these lines we will save the unique id of each of the detected object in a list and then we will find the length of list which gives us the count of Vehicles Entering and Leaving

```
classNames = cococlassNames()
output = cv2.VideoWriter('Output1.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (fr
totalcountup=[]
totalcountdown=[]
limitdown = [309, 407, 598, 407]
limitup = [685, 407, 919, 407]
while True:
    xywh_bboxs = []
```

```
        confs = []
        oids = []
        outputs = []
        ret, frame = cap.read()
        if ret:
          result = list(model.predict(frame, conf=0.5))[0]
          bbox_xyxys = result.prediction.bboxes_xyxy.tolist()
          confidences = result.prediction.confidence
          labels = result.prediction.labels.tolist()
          cv2.line(frame, (limitup[0], limitup[1]), (limitup[2], limitup[3]), (255, 0,0), 2)
          cv2.line(frame, (limitdown[0], limitdown[1]), (limitdown[2], limitdown[3]), (255,0,0
          for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
              bbox = np.array(bbox_xyxy)
              x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
              x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
              conf = math.ceil((confidence*100))/100
              cx, cy = int((x1+x2)/2), int((y1+y2)/2)
              bbox_width = abs(x1-x2)
              bbox_height = abs(y1-y2)
              xcycwh = [cx, cy, bbox_width, bbox_height]
              xywh_bboxs.append(xcycwh)
              confs.append(conf)
              oids.append(int(cls))
          xywhs = torch.tensor(xywh_bboxs)
          confss= torch.tensor(confs)
          outputs = deepsort.update(xywhs, confss, oids, frame)
          if len(outputs)>0:
              bbox_xyxy = outputs[:,:4]
              identities = outputs[:, -2]
              object_id = outputs[:, -1]
              draw_boxes_vehicles_counting_el(frame, bbox_xyxy, identities, object_id)
          output.write(frame)
        else:
            break

    output.release()
    cap.release()
```

```
[2023-09-27 06:50:51] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:50:51] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:50:51] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:50:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:50:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:50:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:50:53] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:50:53] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:50:53] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
```

```
[2023-09-27 06:50:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:50:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:55] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:55] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:56] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:56] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:57] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:57] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-27 06:50:58] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
Total Count Down 1
[2023-09-27 06:50:58] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
Total Count Down 1
[2023-09-27 06:50:59] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
Total Count Down 1
```

3.E. Display the Output Video

```
!rm '/content/result_compressed.mp4'
```

```
input_video_path = '/content/Output1.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```

```
Total Count Up 4
Total Count Down 7
```

3.F. Test on Demo Video 2

3.F.1 Load the Video and Calculate the Frame Width and Frame Height

```
video_path = '/content/NewVehiclesEnteringandLeaving.mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

3.F.2 Creating the Helping Functions

```
#Creating a helper function

def draw_boxes_vehicles_counting_el2(img, bbox, identities=None, categories=None, names=No
    for i, box in enumerate(bbox):
        x1, y1,  x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[0]
        y2 += offset[0]
        cat = int(categories[i]) if categories is not None else 0
        id = int(identities[i]) if identities is not None else 0
        cx, cy = int((x1+x2)/2), int((y1+y2)/2)
        #cv2.circle(frame, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
        #Create Bounding Boxes around the Detected Objects
        cv2.rectangle(img, (x1, y1), (x2, y2), color= compute_color_for_labels(cat),thickne
        label = str(id) + ":" + classNames[cat]
        (w,h), _ = cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, th
        #Create a rectangle above the detected object and add label and confidence score
        t_size=cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, thickne
        c2=x1+t_size[0], y1-t_size[1]-3
        cv2.rectangle(img, (x1, y1), c2, color=compute_color_for_labels(cat), thickness=-1
        cv2.putText(img, str(label), (x1, y1-2), 0, 1/2, [255, 255, 255], thickness=1, line
```

```
            if limitup[0] < cx < limitup[2] and limitup[1] -15 < cy < limitup[3] + 15:
              if totalcountup.count(id)==0:
                totalcountup.append(id)
                cv2.line(img, (limitup[0], limitup[1]), (limitup[2], limitup[3]), (0,255, 0), 
            if limitdown[0] < cx < limitdown[2] and limitdown[1] -15 < cy < limitdown[3] + 15:
              if totalcountdown.count(id)==0:
                totalcountdown.append(id)
                cv2.line(img, (limitdown[0], limitdown[1]), (limitdown[2], limitdown[3]), (0,2
        cv2.rectangle(frame, (1267, 65), (1700, 97), [255, 0, 255], -1, cv2.LINE_AA)
        cv2.putText(frame, str("Vehicle Entering") + ":" + str(len(totalcountup)), (1317, 91),
        cv2.rectangle(frame, (100, 65), (470, 97), [0, 28, 136], -1, cv2.LINE_AA)
        cv2.putText(frame, str("Vehicle Leaving") + ":" + str(len(totalcountdown)), (141, 91),
        print("Total Count Up",len(totalcountup))
        print("Total Count Down",len(totalcountdown))
        return img
```

3.F.3 Vehicles Counting

```
    classNames = cococlassNames()
    output = cv2.VideoWriter('Output1.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (fr
    totalcountup=[]
    totalcountdown=[]
    limitdown = [225, 850, 963, 850]
    limitup = [979, 850, 1667, 850]
    while True:
        xywh_bboxs = []
        confs = []
        oids = []
        outputs = []
        ret, frame = cap.read()
        if ret:
          result = list(model.predict(frame, conf=0.40))[0]
          bbox_xyxys = result.prediction.bboxes_xyxy.tolist()
          confidences = result.prediction.confidence
          labels = result.prediction.labels.tolist()
          cv2.line(frame, (limitup[0], limitup[1]), (limitup[2], limitup[3]), (255, 0,0), 2)
          cv2.line(frame, (limitdown[0], limitdown[1]), (limitdown[2], limitdown[3]), (255,0,0
          for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
              bbox = np.array(bbox_xyxy)
              x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
              x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
              conf = math.ceil((confidence*100))/100
              cx, cy = int((x1+x2)/2), int((y1+y2)/2)
              bbox_width = abs(x1-x2)
              bbox_height = abs(y1-y2)
              xcycwh = [cx, cy, bbox_width, bbox_height]
              xywh_bboxs.append(xcycwh)
              confs.append(conf)
              oids.append(int(cls))
          xywhs = torch.tensor(xywh_bboxs)
          confss= torch.tensor(confs)
          outputs = deepsort.update(xywhs, confss, oids, frame)
          if len(outputs)>0:
              bbox_xyxy = outputs[:,:4]
              identities = outputs[:, -2]
```

```
            object_id = outputs[:, -1]
            draw_boxes_vehicles_counting_el2(frame, bbox_xyxy, identities, object_id)
        output.write(frame)
    else:
        break

output.release()
cap.release()
```

```
[2023-09-27 06:52:16] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 06:52:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:52:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:52:18] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:52:18] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:52:19] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 06:52:19] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
Total Count Down 0
[2023-09-27 06:52:20] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 2
Total Count Down 1
[2023-09-27 06:52:20] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 1
[2023-09-27 06:52:20] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 1
[2023-09-27 06:52:21] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 2
[2023-09-27 06:52:21] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 2
[2023-09-27 06:52:21] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 3
[2023-09-27 06:52:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 3
[2023-09-27 06:52:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 4
[2023-09-27 06:52:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 4
[2023-09-27 06:52:23] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 5
[2023-09-27 06:52:23] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 5
[2023-09-27 06:52:24] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

```
Total Count Up 5
Total Count Down 5
[2023-09-27 06:52:24] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 5
```

3.F.4 Display the Output Video

```
!rm '/content/result_compressed.mp4'
```

```
input_video_path = '/content/Output1.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```

```
Total Count Up 117
Total Count Down 108
```

4. Vehicles Counting (Entering and Leaving) with YOLO-NAS and SORT Object Tracking

4.A. Dowload the SORT File

Previously we have integrated YOLO-NAS with DeepSORT to implement Object Tracking, now we will intergrate YOLO-NAS with SORT to implement Object Tracking. SORT is an approach to object tracking where Kalman filters and Hungarian algorithms are used to track objects. SORT consists of four key components which includes detection, estimation, data association and creation & deletion of track identities. SORT algorithm performs very well in terms of tracking precision and accuracy

```
!gdown "https://drive.google.com/uc?id=1AhwiIb2umnJpZwunbfCKiyjanzjX_xfx&confirm=t"
```

```
Downloading...
From: https://drive.google.com/uc?id=1AhwiIb2umnJpZwunbfCKiyjanzjX_xfx&confirm=t
To: /content/sort.py
100% 13.9k/13.9k [00:00<00:00, 58.4MB/s]
```

4.B. Import All the Required Libraries

Importing all the functions, classes, and variables from a module named "sort", which helps us to directly access and use any functions, classes, or variables defined in the "sort" module without having to prefix them with the module name.

```
from sort import *
```

4.C. Load the Input Video and Calculate the Frame Width and Frame Height

```
video_path = '/content/NewVehiclesEnteringandLeaving.mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

4.D. Download the Pre-Trained YOLO-NAS Model Weights

```
model_name = 'yolo_nas_l'
model = load_model(model_name)
```

```
[2023-09-27 07:00:15] INFO - checkpoint_utils.py - License Notification: YOLO-NAS pre-traine
https://github.com/Deci-AI/super-gradients/blob/master/LICENSE.YOLONAS.md
By downloading the pre-trained weight files you agree to comply with these terms.
[2023-09-27 07:00:15] INFO - checkpoint_utils.py - Successfully loaded pretrained weights fo
```

4.E. Creating the Color Palette

```
classNames = cococlassNames()
colors = [[random.randint(0, 255) for _ in range(3)]
          for _ in range(len(classNames))]
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
```

4.F. Test on Demo Video 1

The code snippet below will implement vechicles counting, we will create two dummy lines one to count the vehicles entering and one to count the vehicles leaving and when the vehicles intersect with any of these lines we will save the unique id of each of the detected object in a list and then we will find the length of list which gives us the count of Vehicles Entering and Leaving

```
classNames = cococlassNames()
out = cv2.VideoWriter('Output3.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (frame
totalcountup=[]
totalcountdown=[]
limitdown = [225, 850, 963, 850]
limitup = [979, 850, 1667, 850]
tracker = Sort(max_age = 20, min_hits=3, iou_threshold=0.3)
count=0
while True:
    ret, frame = cap.read()
    count += 1
    if ret:
        detections = np.empty((0,6))
        result = list(model.predict(frame, conf=0.40))[0]
        bbox_xyxys = result.prediction.bboxes_xyxy.tolist()
        confidences = result.prediction.confidence
        labels = result.prediction.labels.tolist()
        cv2.line(frame, (limitup[0], limitup[1]), (limitup[2], limitup[3]), (255, 0,0), 2)
        cv2.line(frame, (limitdown[0], limitdown[1]), (limitdown[2], limitdown[3]), (255,0,
        for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
            bbox = np.array(bbox_xyxy)
            x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            classname = int(cls)
            class_name = classNames[classname]
            conf = math.ceil((confidence*100))/100
            currentArray = np.array([x1, y1, x2, y2, conf, cls])
            detections = np.vstack((detections, currentArray))
        tracker_dets = tracker.update(detections)
        if len(tracker_dets) >0:
            bbox_xyxy = tracker_dets[:,:4]
            identities = tracker_dets[:, 8]
            categories = tracker_dets[:, 4]
            draw_boxes_vehicles_counting_el2(frame, bbox_xyxy, identities, categories)
        out.write(frame)
    else:
        break

out.release()
cap.release()
```

```
[2023-09-27 07:01:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:01:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 07:01:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-27 07:01:53] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
Total Count Down 0
[2023-09-27 07:01:53] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
Total Count Down 0
[2023-09-27 07:01:53] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 2
Total Count Down 1
[2023-09-27 07:01:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

```
Total Count Up 3
Total Count Down 1
[2023-09-27 07:01:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 1
[2023-09-27 07:01:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 2
[2023-09-27 07:01:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 2
[2023-09-27 07:01:55] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 3
[2023-09-27 07:01:55] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 3
Total Count Down 4
[2023-09-27 07:01:56] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 4
[2023-09-27 07:01:56] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 5
[2023-09-27 07:01:57] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 5
[2023-09-27 07:01:57] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 4
Total Count Down 5
[2023-09-27 07:01:57] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 5
Total Count Down 5
[2023-09-27 07:01:58] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 5
Total Count Down 5
[2023-09-27 07:01:58] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 5
Total Count Down 7
[2023-09-27 07:01:59] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 5
```

4.G. Display the Output Video

```
!rm '/content/result_compressed.mp4'
```

```
input_video_path = '/content/Output3.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```

4.H. Test on Demo Video 2

```
video_path = '/content/test3.mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

```
classNames = cococlassNames()
out = cv2.VideoWriter('Output3.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (frame
totalcountup=[]
totalcountdown=[]
limitdown = [309, 407, 598, 407]
limitup = [685, 407, 919, 407]
tracker = Sort(max_age = 20, min_hits=3, iou_threshold=0.3)
count=0
while True:
    ret, frame = cap.read()
    count += 1
    if ret:
        detections = np.empty((0,6))
        result = list(model.predict(frame, conf=0.5))[0]
        bbox_xyxys = result.prediction.bboxes_xyxy.tolist()
        confidences = result.prediction.confidence

        labels = result.prediction.labels.tolist()
        cv2.line(frame, (limitup[0], limitup[1]), (limitup[2], limitup[3]), (255, 0,0), 2)
        cv2.line(frame, (limitdown[0], limitdown[1]), (limitdown[2], limitdown[3]), (255,0,
        for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
            bbox = np.array(bbox_xyxy)
            x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            classname = int(cls)
            class_name = classNames[classname]
            conf = math.ceil((confidence*100))/100
            currentArray = np.array([x1, y1, x2, y2, conf, cls])
            detections = np.vstack((detections, currentArray))
        tracker_dets = tracker.update(detections)
        if len(tracker_dets) >0:
            bbox_xyxy = tracker_dets[:,:4]
            identities = tracker_dets[:, 8]
            categories = tracker_dets[:, 4]
```

```
                draw_boxes_vehicles_counting_el(frame, bbox_xyxy, identities, categories)
            out.write(frame)
        else:
            break

    out.release()
    cap.release()
```

```
[2023-09-23 13:50:25] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-23 13:50:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:28] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:28] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 0
[2023-09-23 13:50:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 0
Total Count Down 1
[2023-09-23 13:50:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

```
Total Count Up 1
Total Count Down 1
[2023-09-23 13:50:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
Total Count Up 1
```

4.I. Display the Output Video

```
!rm '/content/result_compressed.mp4'
```

```
input_video_path = '/content/Output3.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```

5. YOLO-NAS for object tracking on a custom dataset (e.g., ship detection).

We have seen how we can do object tracking with YOLO-NAS and SORT / DeepSORT algorithms. In the previous examples we are using pre-trained weights from the COCO dataset. Now we will go one step ahead and we will implement object tracking with YOLO-NAS and DeepSORT on a custom dataset. We will first train the YOLO-NAS model on a ship's dataset, available publicly on roboflow. After training the YOLO-NAS model on the ship's dataset we will integrate DeepSORT object tracking with our custom trained YOLO-NAS model. The code snippet is provided below.

5.A. Download the Custom Model Weights and the Input Videos

```
!gdown "https://drive.google.com/uc?id=16WRUKxJLhAt_XAbAy22mS3YhyYqt7KDF&confirm=t"
!gdown "https://drive.google.com/uc?id=1M5cgfvz-xG6BxCYVAs8LekMpGxu14LVY&confirm=t"
!gdown "https://drive.google.com/uc?id=104hjeM0nMFT_fJFXsPwmnYQ-sQLEmKsJ&confirm=t"
```

```
Downloading...
From: https://drive.google.com/uc?id=16WRUKxJLhAt_XAbAy22mS3YhyYqt7KDF&confirm=t
To: /content/ckpt_best.pth
100% 256M/256M [00:03<00:00, 80.6MB/s]
Downloading...
From: https://drive.google.com/uc?id=1M5cgfvz-xG6BxCYVAs8LekMpGxu14LVY&confirm=t
To: /content/ship1.mp4
100% 16.6M/16.6M [00:00<00:00, 68.1MB/s]
Downloading...
From: https://drive.google.com/uc?id=104hjeM0nMFT_fJFXsPwmnYQ-sQLEmKsJ&confirm=t
To: /content/ship2.mp4
100% 2.53M/2.53M [00:00<00:00, 167MB/s]
```

5.B. Load the input video and calculate the Frame Width and Frame Height

```
video_path = '/content/ship2.mp4'
cap, frame_width, frame_height = get_video_info(video_path)
```

5.C. Initialize the Custom Model Weights

```
device = torch.device("cuda:0") if torch.cuda.is_available() else torch.device("cpu")
model = models.get('yolo_nas_s', num_classes= 1, checkpoint_path='/content/ckpt_best.pth')
```

```
[2023-09-27 07:05:37] INFO - checkpoint_utils.py - Successfully loaded model weights from /c
```

5.D. Initialize the DeepSORT

```
deepsort = initialize_deepsort()
```

```
[2023-09-27 07:06:32] INFO - feature_extractor.py - Loading weights from deep_sort_pytorch/c
```

5.E. Creating a Color Palette

```
#Creating a helper function

def draw_boxes(img, bbox, identities=None, categories=None, names=None, offset=(0,0)):
    for i, box in enumerate(bbox):
        x1, y1,  x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[0]
        y2 += offset[0]
```

```
            cat = int(categories[i]) if categories is not None else 0
            id = int(identities[i]) if identities is not None else 0
            #Create Bounding Boxes around the Detected Objects
            cv2.rectangle(img, (x1, y1), (x2, y2), color= compute_color_for_labels(cat),thickne
            label = str(id) + ":" + classNames[cat]
            (w,h), _ = cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, th
            #Create a rectangle above the detected object and add label and confidence score
            t_size=cv2.getTextSize(str(label), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1/2, thickne
            c2=x1+t_size[0], y1-t_size[1]-3
            cv2.rectangle(frame, (x1, y1), c2, color=compute_color_for_labels(cat), thickness=-
            cv2.putText(frame, str(label), (x1, y1-2), 0, 1/2, [255, 255, 255], thickness=1, l
        return img
```

```
classNames = ["Boat"]
colors = [[random.randint(0, 255) for _ in range(3)]
          for _ in range(len(classNames))]
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
```

```
def compute_color_for_labels(label):
    """
    Simple function that adds fixed color depending on the class
    """
    if label == 0:
        color = (85, 45, 255)
    return tuple(color)
```

5.F. Object Tracking with YOLO-NAS and DeepSORT on Custom Dataset

```
classNames = ['Boat']
output = cv2.VideoWriter('Output1.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (fr

while True:
    xywh_bboxs = []
    confs = []
    oids = []
    outputs = []
    ret, frame = cap.read()
    if ret:
      result = list(model.predict(frame, conf=0.55))[0]
      bbox_xyxys = result.prediction.bboxes_xyxy.tolist()
      confidences = result.prediction.confidence
      labels = result.prediction.labels.tolist()
      for (bbox_xyxy, confidence, cls) in zip(bbox_xyxys, confidences, labels):
          bbox = np.array(bbox_xyxy)
          x1, y1, x2, y2 = bbox[0], bbox[1], bbox[2], bbox[3]
          x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
          conf = math.ceil((confidence*100))/100
          cx, cy = int((x1+x2)/2), int((y1+y2)/2)
          bbox_width = abs(x1-x2)
          bbox_height = abs(y1-y2)
          xcycwh = [cx, cy, bbox_width, bbox_height]
```

```
            xywh_bboxs.append(xcycwh)
            confs.append(conf)
            oids.append(int(cls))
        xywhs = torch.tensor(xywh_bboxs)
        confss= torch.tensor(confs)
        outputs = deepsort.update(xywhs, confss, oids, frame)
        if len(outputs)>0:
            bbox_xyxy = outputs[:,:4]
            identities = outputs[:, -2]
            object_id = outputs[:, -1]
            draw_boxes(frame, bbox_xyxy, identities, object_id)
        output.write(frame)
      else:
          break

output.release()
cap.release()
```

```
[2023-09-27 07:06:40] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:41] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:42] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:43] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:45] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:46] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:46] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:47] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:48] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:49] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:50] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:51] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:52] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:54] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:55] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:57] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:58] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:06:59] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:00] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:01] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:02] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:02] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:03] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:04] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:05] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:06] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:07] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:09] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:10] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:12] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:13] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:14] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:15] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:16] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:17] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:18] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:19] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:20] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:21] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:22] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:23] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:24] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

```
[2023-09-27 07:07:26] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:27] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:28] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:29] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:30] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:31] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:32] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:33] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:34] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:35] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:36] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:37] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:38] INFO - pipelines.py - Fusing some of the model's layers. If this tak
[2023-09-27 07:07:39] INFO - pipelines.py - Fusing some of the model's layers. If this tak
```

5.G. Display the Output Video

```
!rm '/content/result_compressed.mp4'
```

```
rm: cannot remove '/content/result_compressed.mp4': No such file or directory
```

```
input_video_path = '/content/Output1.avi'  # Replace with your input video path
display_compressed_video(input_video_path)
```