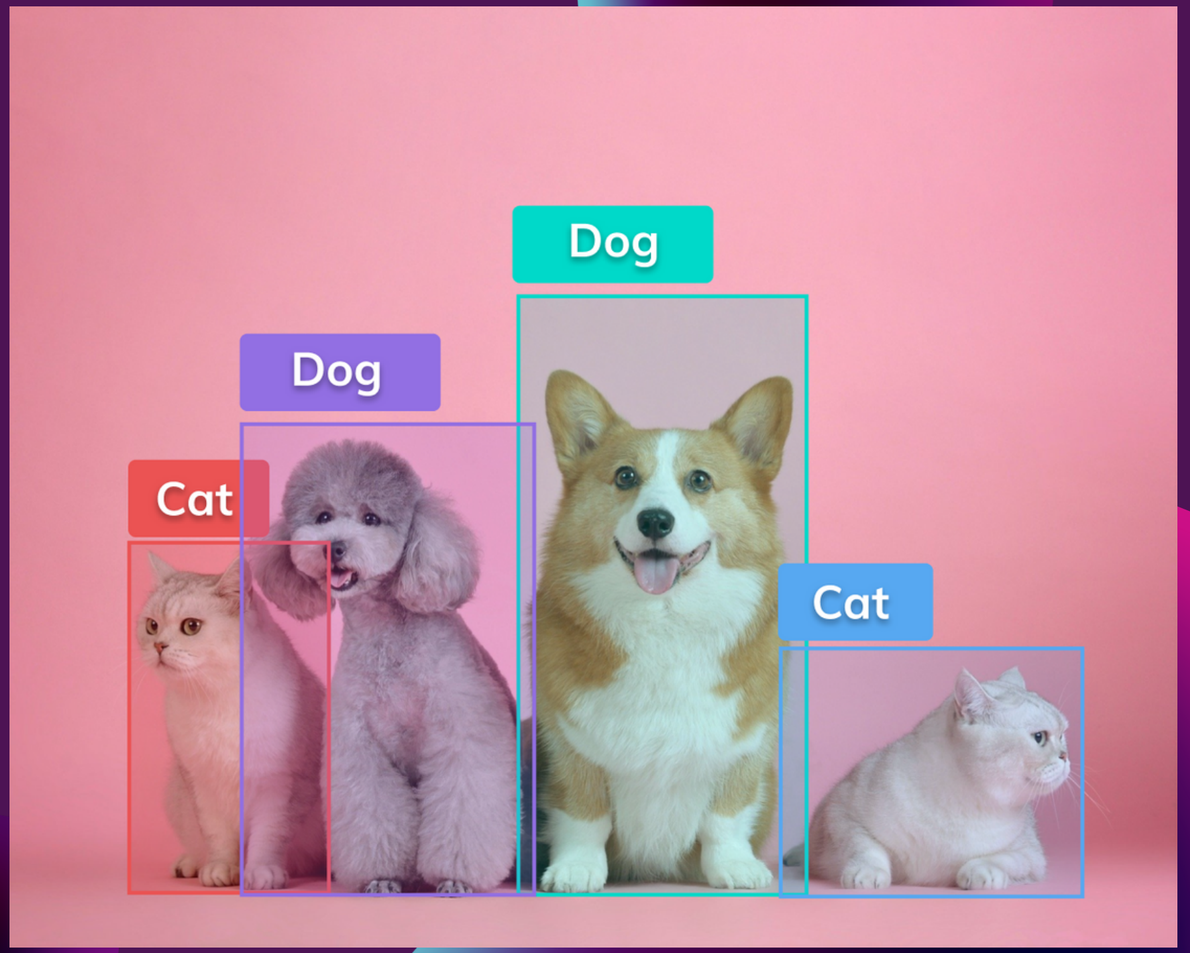
Vehicle counting and classification

Oprea Olivia Maria
Mitulescu Alexandra

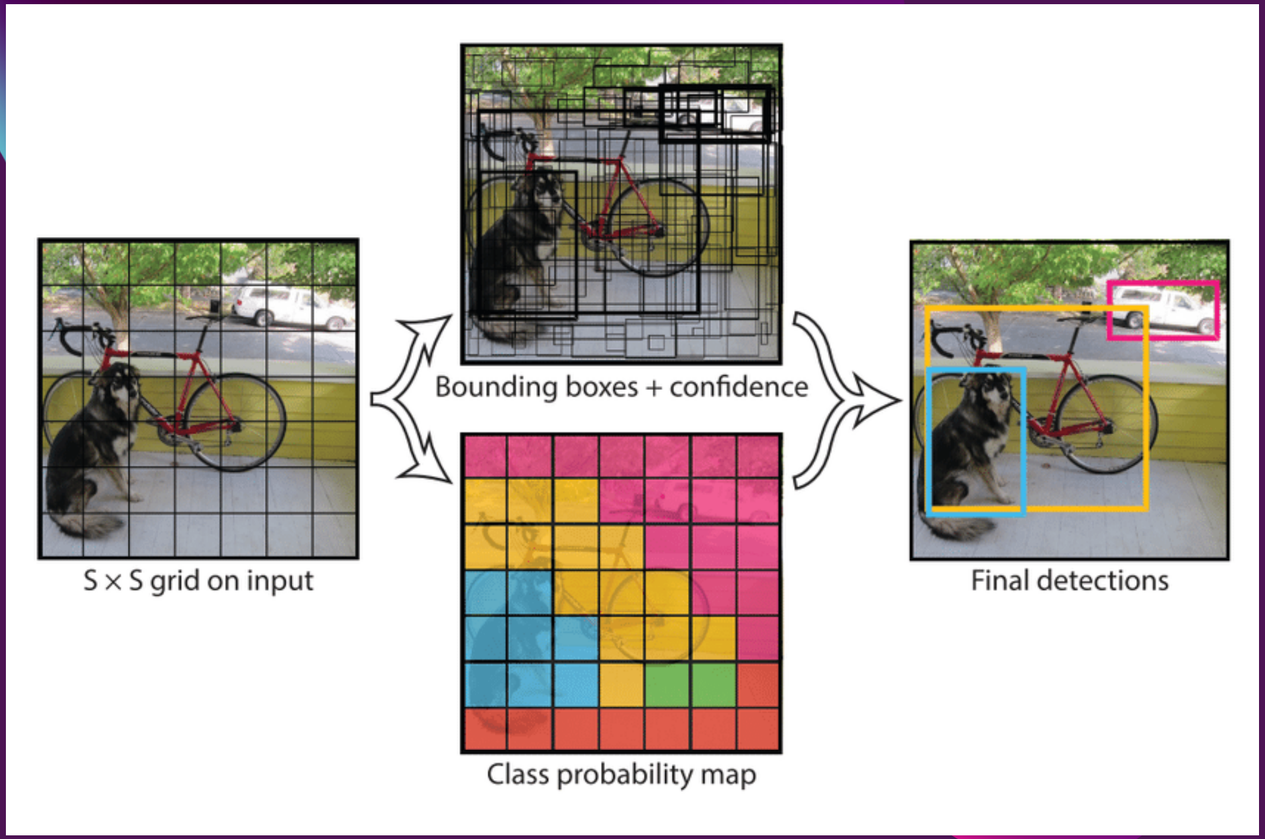
Object Detection and Classification

Object detection is a computer vision technique for locating instances of semantic objects in digital images or videos. This task involves identifying the location of an object in an image and drawing a bounding box around it.

Object classification is a computer vision task that involves assigning a class label to an object in an image based on its features and characteristics. The goal is to identify the type of object in the image, such as a car, truck, or bicycle.



Vehicle Counting and Classification using YOLO



YOLO (You Only Look Once) is an object detection algorithm that uses a single convolutional neural network (CNN) to perform object detection and classification in real-time. YOLO divides an image into multiple grid cells and predicts the presence of an object and its class in each grid cell.

Paper: Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao - YOLOv4: Optimal Speed and Accuracy of Object Detection (2020)

Overview of YOLO architecture:

The YOLO architecture consists of two parts: feature extraction and prediction. The feature extraction part uses multiple convolutional and max pooling layers to extract features from the input image. The prediction part consists of several fully connected layers that predict the bounding boxes and class probabilities. The architecture is designed to be fast and efficient, allowing real-time object detection on high-resolution images.

To use YOLO for vehicle counting and classification, the following pre-processing steps are required:

- Image acquisition: Capturing high-quality images of vehicles in various environments, angles and scales.
- Image annotation: Marking the objects of interest in the images, i.e., vehicles, to train the YOLO model.
- Data pre-processing: Resizing and normalizing the images to a specific size, splitting the dataset into training, validation, and testing sets, and encoding the object classes.

The steps involved in YOLO object detection and classification are:

- Model training: Training the YOLO model on the annotated images using a deep learning framework like TensorFlow or PyTorch.
- Model evaluation: Evaluating the model on the validation set to measure its performance and make improvements if necessary.
- Model inference: Using the trained model to perform object detection and classification on new images.
- Post-processing: Performing any additional operations on the output of the YOLO model, such as non-maximum suppression to eliminate overlapping bounding boxes, or filtering the results based on confidence scores.

In the context of vehicle counting and classification, YOLO is an ideal choice for a number of reasons:

- Real-time processing: YOLO is designed to perform object detection in real-time, making it a suitable choice for applications that require rapid processing of video or image data.
- Single-shot detection: Unlike other object detection algorithms that may require multiple scans or passes over the input data, YOLO performs object detection in a single shot, which reduces computational overhead and makes it faster.
- High accuracy: YOLO has a high accuracy in object detection and classification tasks, making it suitable for vehicle counting and classification applications where accuracy is critical.

Comparison of YOLO with other object detection and classification algorithms:

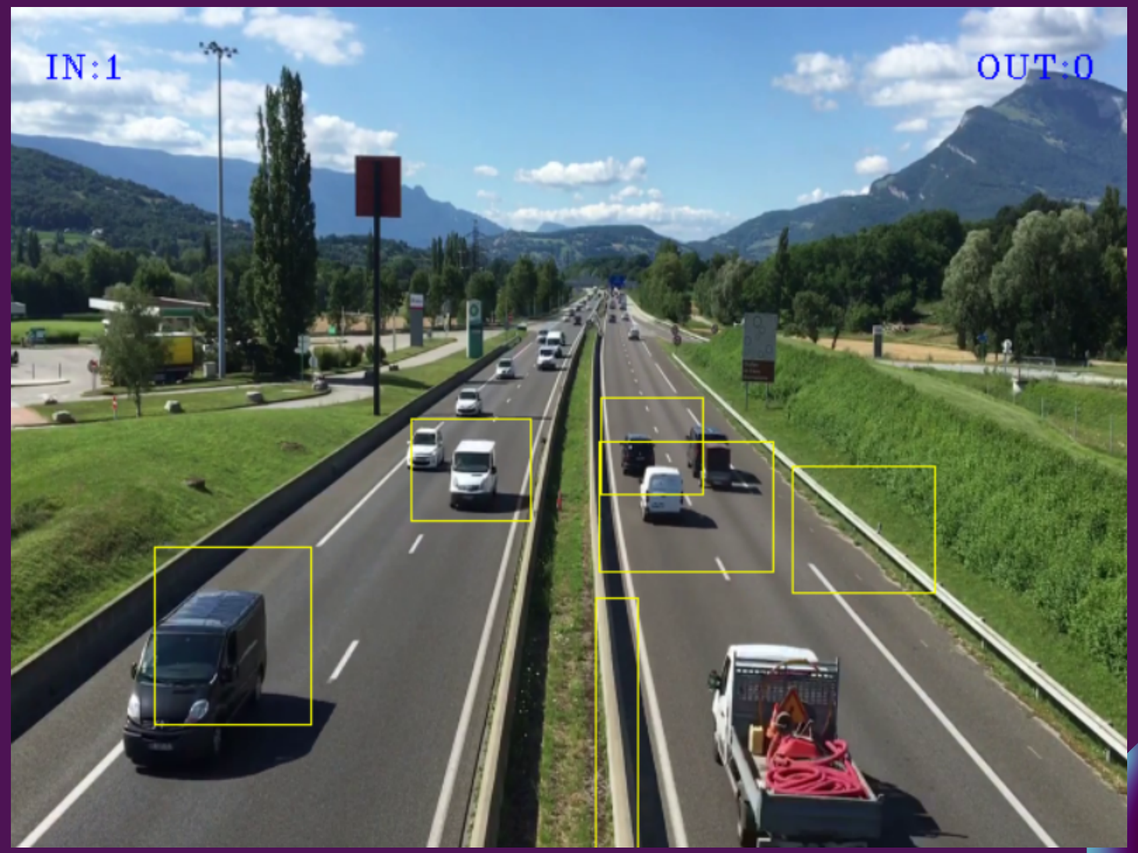
YOLO can be compared to other popular object detection and classification algorithms such as R-CNN, Faster R-CNN, and SSD. Some key differences include:

- Speed: YOLO is one of the fastest object detection algorithms available, which makes it a good choice for real-time applications.
- Complexity: YOLO is relatively simple in terms of architecture and training, which makes it easier to implement and maintain compared to other algorithms.
- Accuracy: While YOLO has a high accuracy in object detection, it may not be as accurate as other algorithms such as R-CNN and Faster R-CNN, which are designed specifically for high accuracy.

The limitations of YOLO for vehicle counting and classification include:

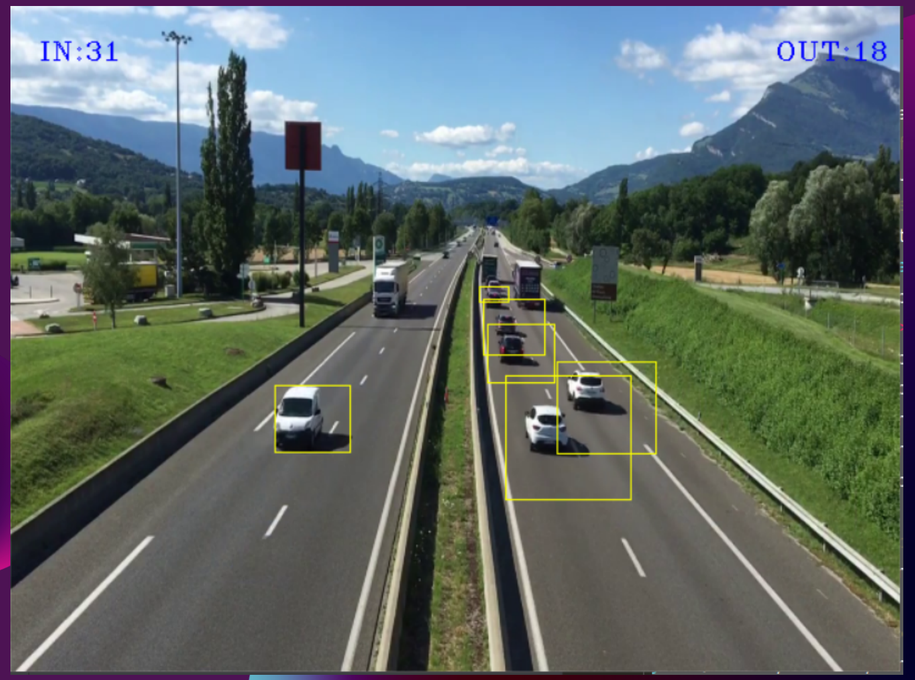
1. High computational cost: YOLO requires a high amount of computational power to run, making it difficult to use on embedded systems and low-end hardware.
2. Detection accuracy: While YOLO has a good trade-off between speed and accuracy, it may not perform as well as other algorithms for detection of smaller objects or objects with similar features.
3. Image size constraints: YOLO is designed to work with larger images, and may not perform as well when used on smaller images.

Output of our code:

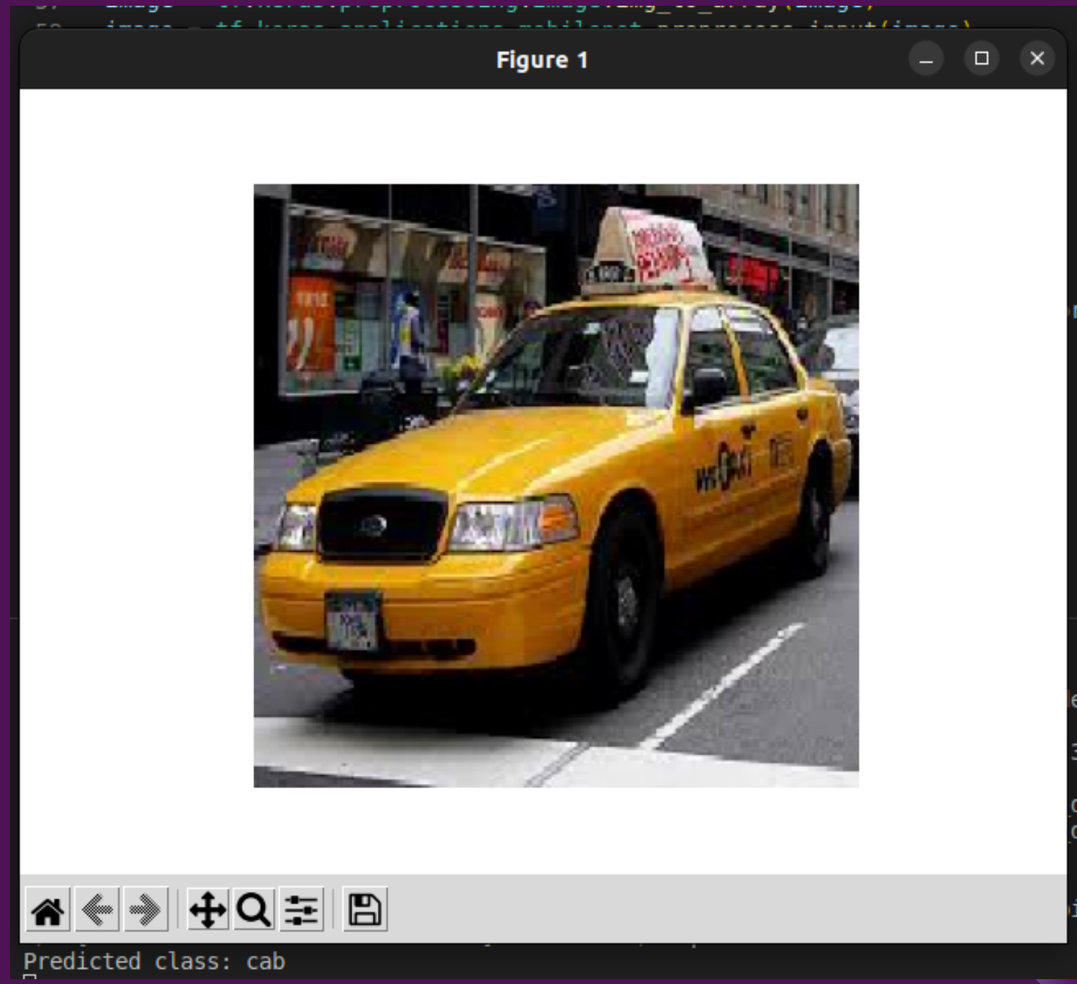


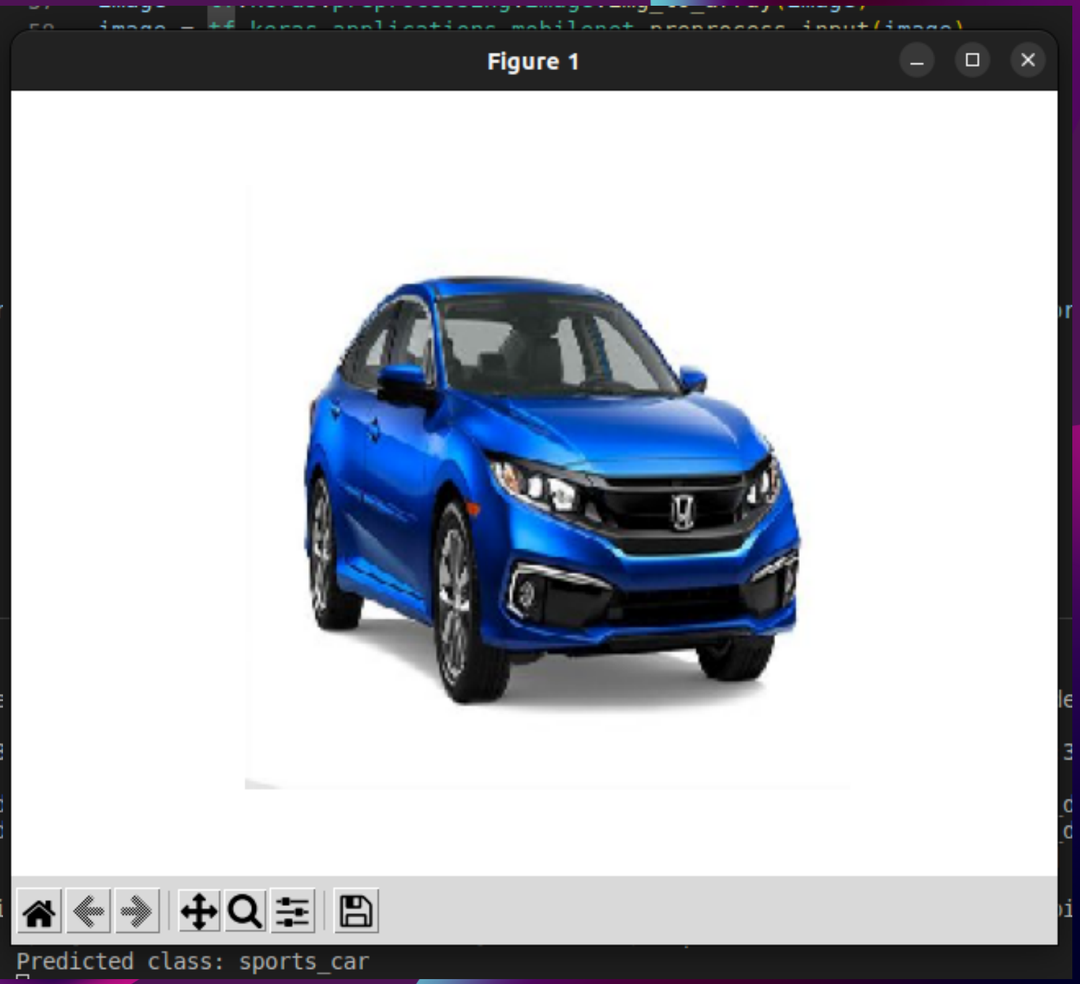

IN:16
OUT:11


IN:21
OUT:13

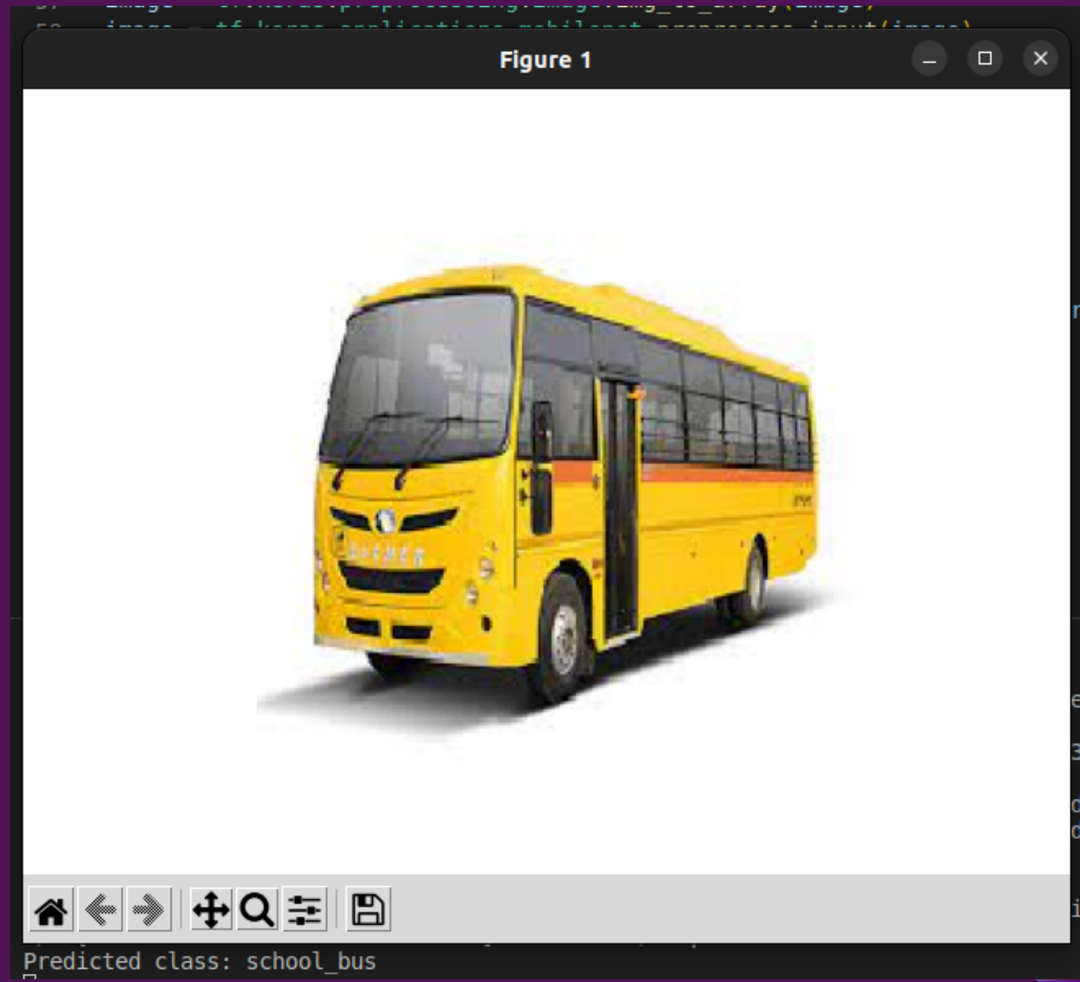

IN:31
OUT:18

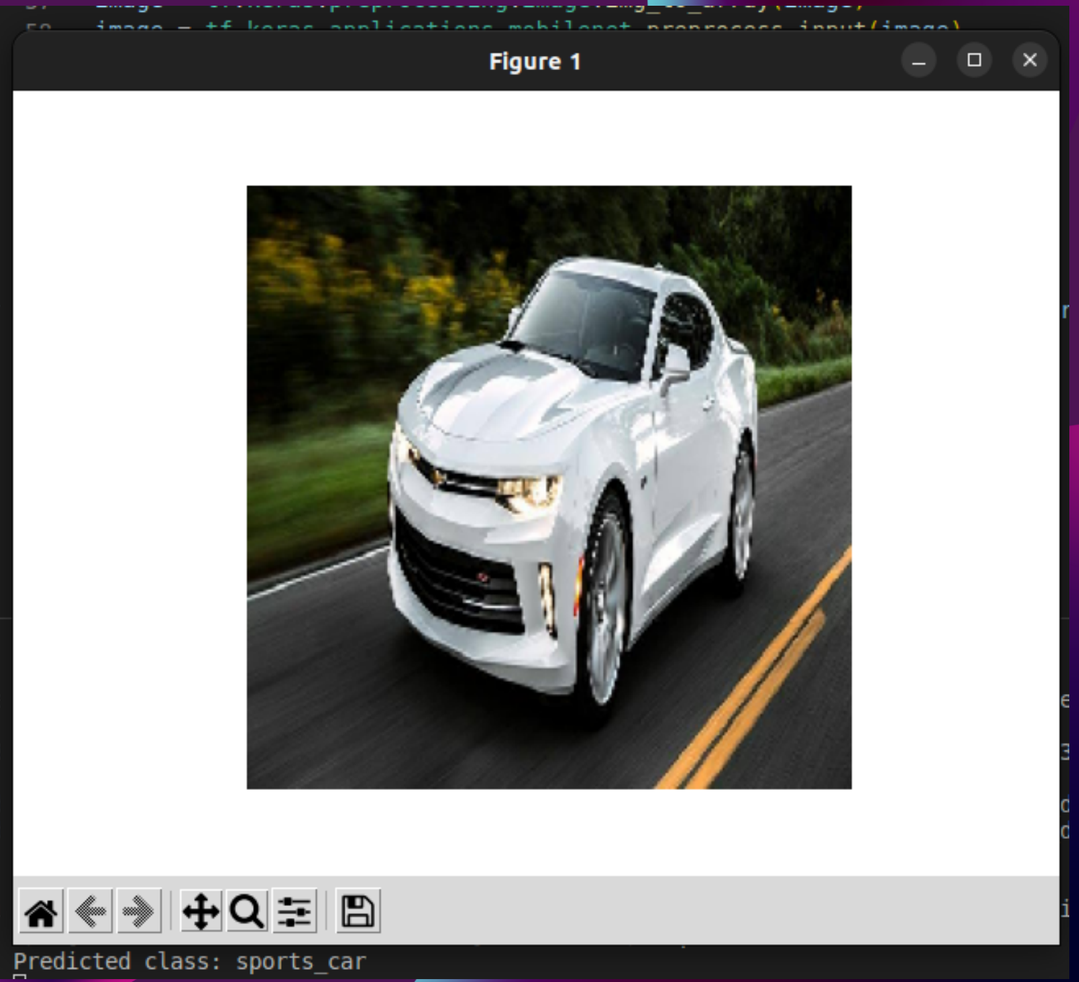
Output of our code:



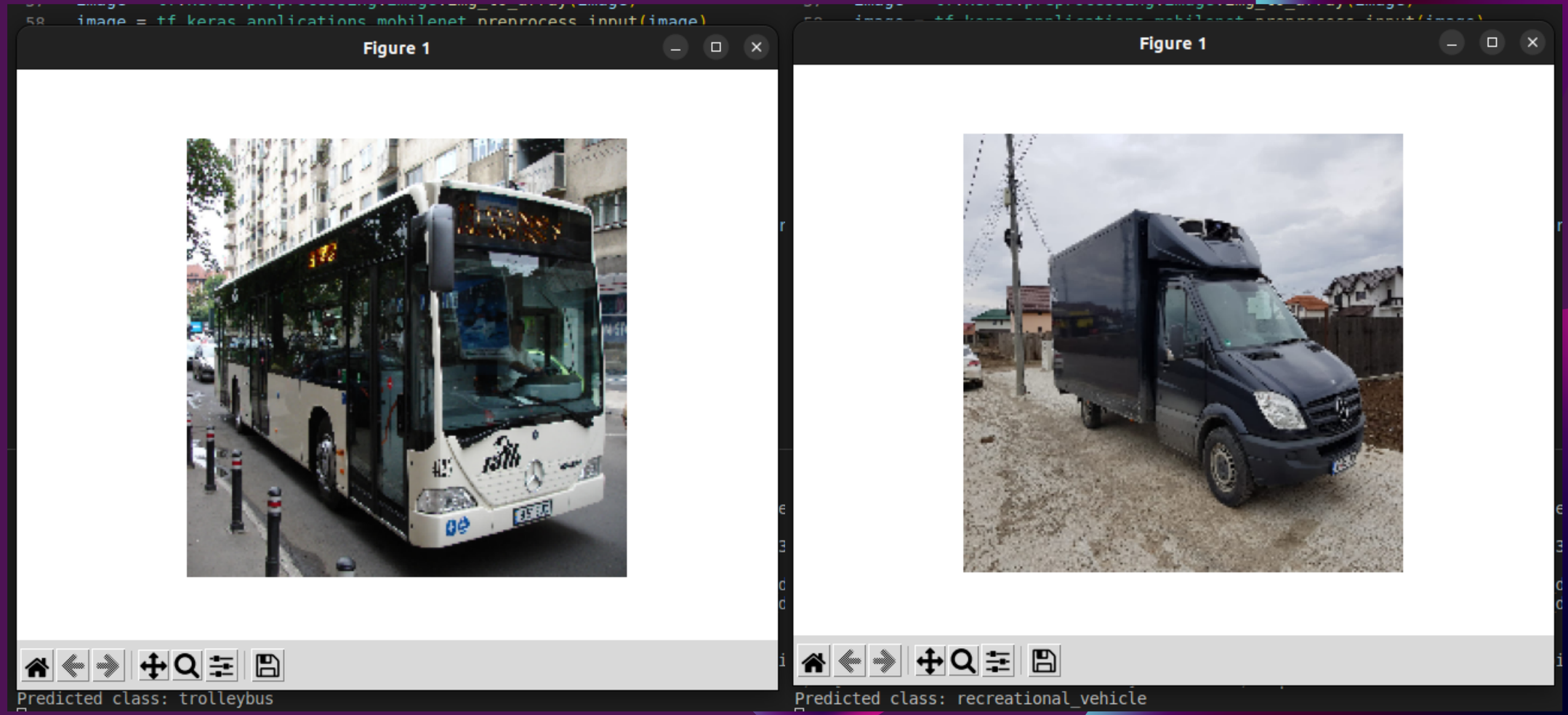


Output of our code:





Output of our code:



Sometimes he is not doing the right things

Conclusion

In conclusion, vehicle counting and classification is an important computer vision task with numerous real-world applications, including traffic management and urban planning. YOLO is a powerful and efficient algorithm that can be used to perform vehicle counting and classification. It has a unique architecture that allows it to detect objects in an image quickly and accurately.

Thank you