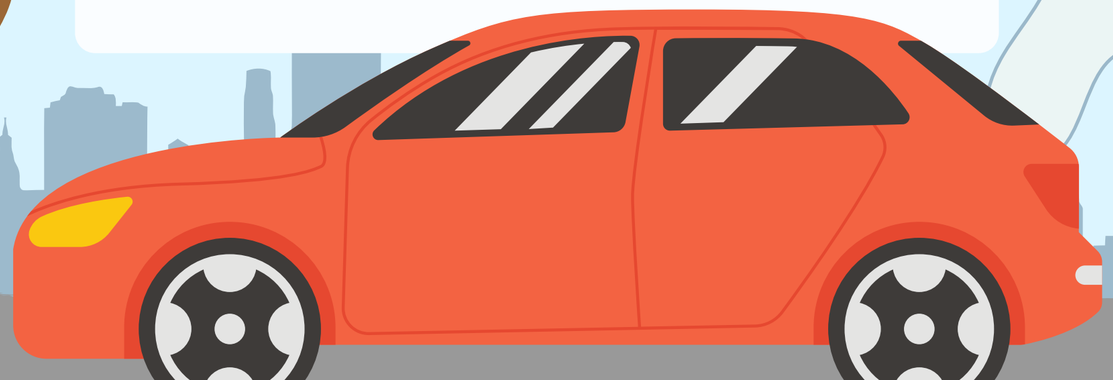
GREEN FLASH

“ Traffic light cars”

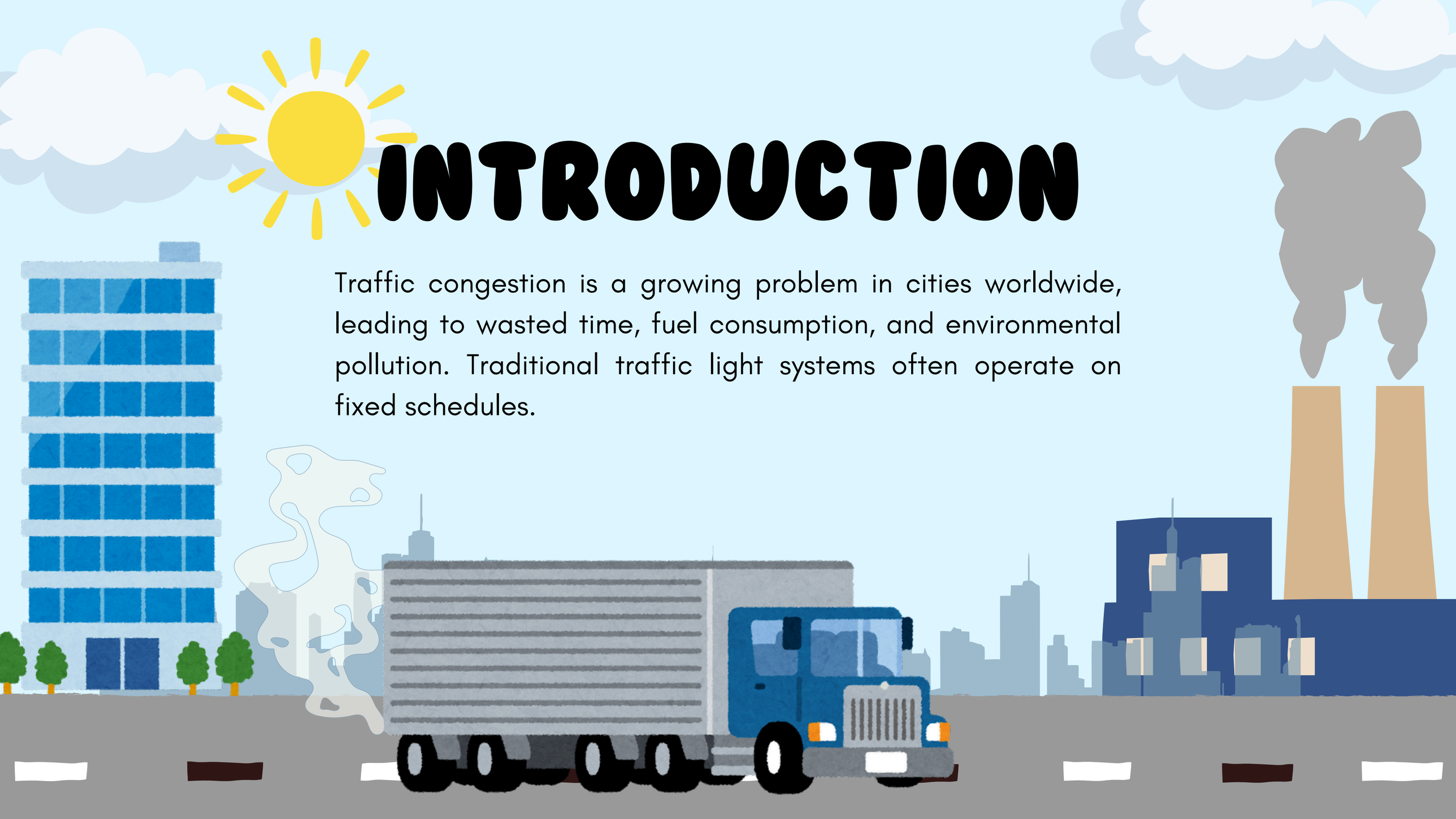
group 4

TABLE OF CONTENTS



INTRODUCTION

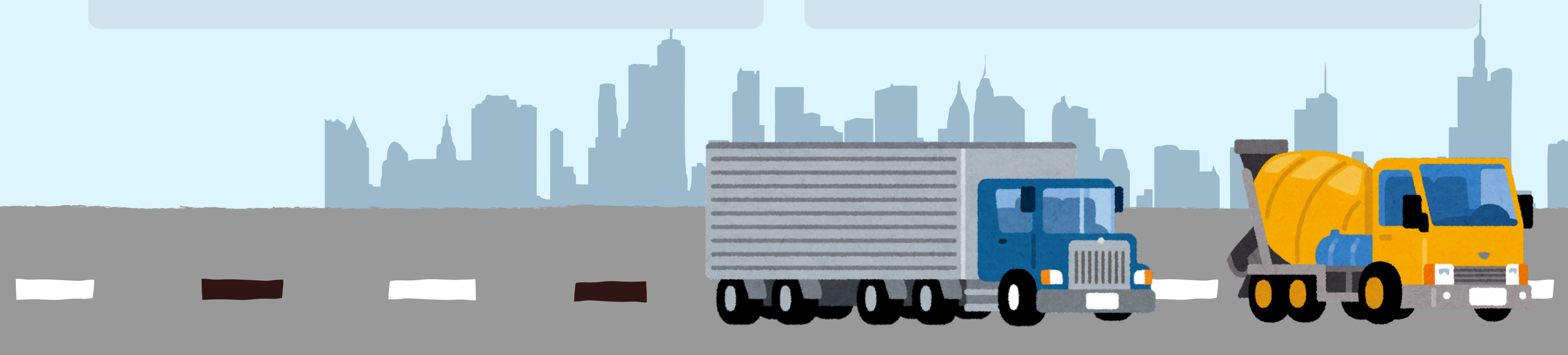
Traffic congestion is a growing problem in cities worldwide, leading to wasted time, fuel consumption, and environmental pollution. Traditional traffic light systems often operate on fixed schedules.



CHALLENGE / SOLUTION

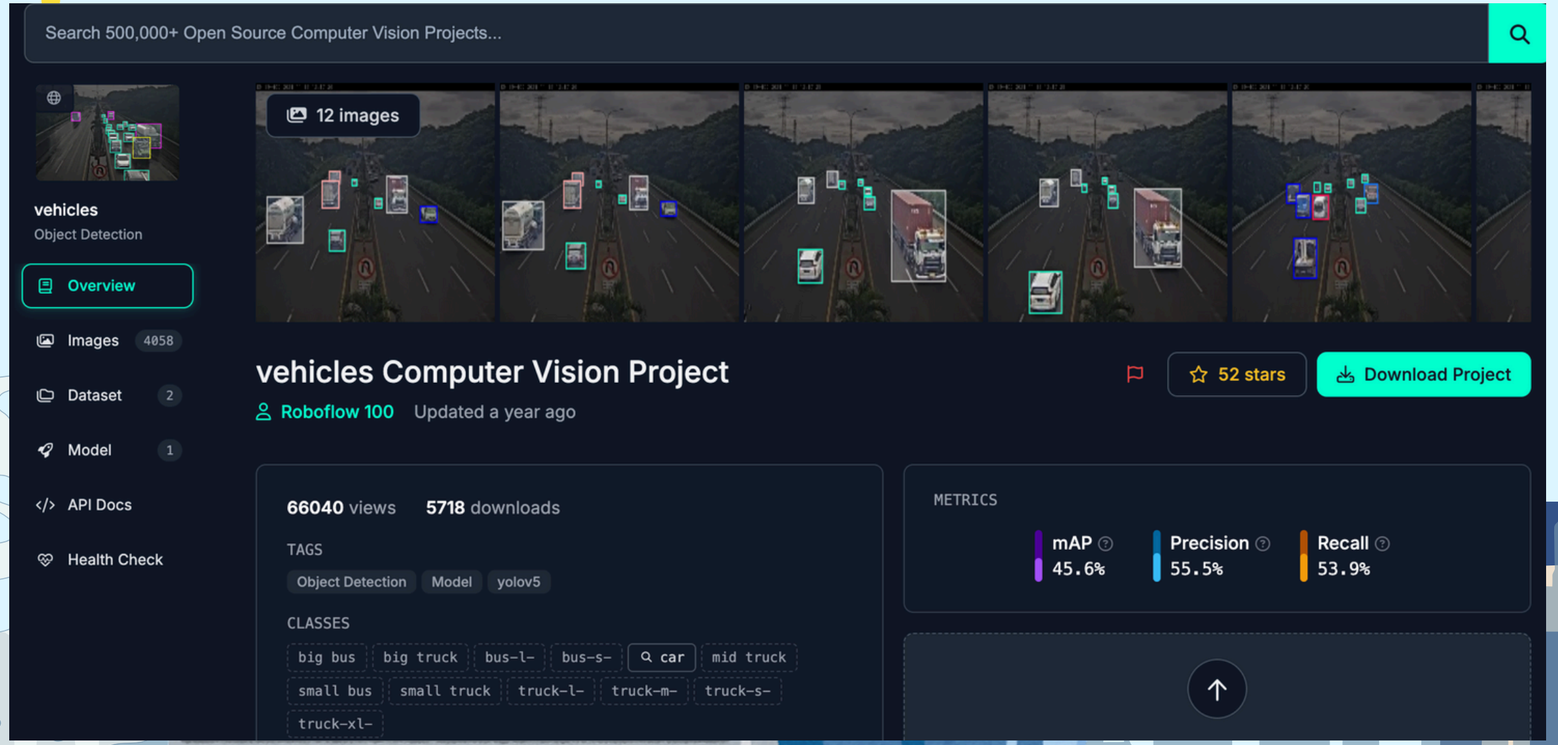
The Challenge:Traditional traffic light systems often operate on fixed schedules, leading to inefficient traffic flow and unnecessary delays.

Our Solution: This project proposes a smart traffic count system that utilizes deep learning to dynamically adjust traffic count timings based on real-time vehicle detection. This system will leverage the power of computer vision



DATASET

FROM ROBOFLOW



METHODOLOGY

```
class CarDetector:
    def __init__(self):
        self.model = fasterrcnn_resnet50_fpn(pretrained=True)
        self.model.eval()

    def detect_cars(self, image_path, labels_dir):
        image = Image.open(image_path)
        image_tensor = F.to_tensor(image)
        # We won't use the model's output, we will use the labels instead
        # outputs = self.model([image_tensor])

        # Load bounding boxes from the labels directory
        label_file = os.path.join(labels_dir, os.path.basename(image_path).replace('.jpg'
        bounding_boxes = []
        if os.path.exists(label_file):
            with open(label_file, 'r') as f:
                for line in f:
                    class_label, x_center, y_center, width, height = map(float, line.spli
                    bounding_boxes.append([class_label, x_center, y_center, width, height
```

METHODOLOGY

```
import numpy as np
import cv2
# Usage example
car_detector = CarDetector()
images_dir = '/content/Car-2/valid/images'
labels_dir = '/content/Car-2/valid/labels'
image_path = '/content/Car-2/valid/images/adit_mp4-1003_jpg.rf.4044e4fdd050b187b388a4f72c
original_image, bounding_boxes = car_detector.detect_cars(image_path, labels_dir)
# Convert the PIL Image to a NumPy array for OpenCV
image_np = np.array(original_image)
# Convert RGB to BGR
image_np = cv2.cvtColor(image_np, cv2.COLOR_RGB2BGR)
# Draw bounding boxes from label files
image_width, image_height = original_image.size
# Counter for the number of cars detected
car_count = 0
for box in bounding_boxes:
    _, x_center, y_center, width, height = box
    x1 = int((x_center - width / 2) * image_width)
    y1 = int((y_center - height / 2) * image_height)
    x2 = int((x_center + width / 2) * image_width)
    y2 = int((y_center + height / 2) * image_height)
    cv2.rectangle(image_np, (x1, y1), (x2, y2), (0, 255, 0), 2)
    car_count += 1
# Print the total number of cars detected
print(f"Number of cars detected: {car_count}")
cv2_imshow(image_np)
```

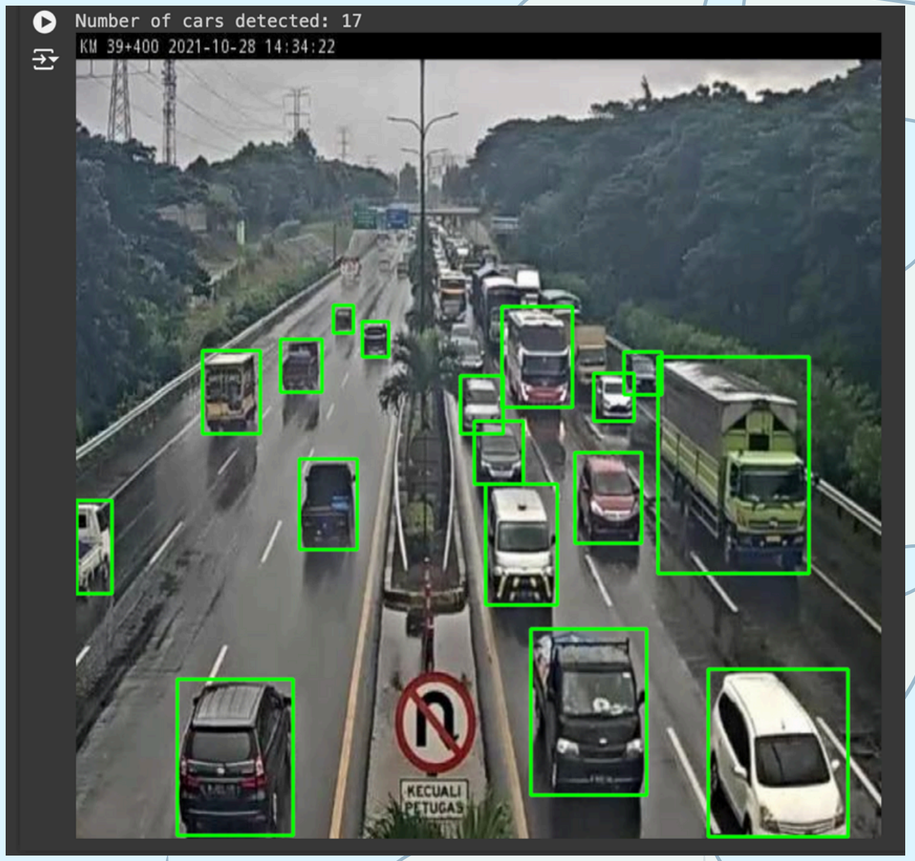
METHODOLOGY

pre-trained Faster R-CNN model with a ResNet-50 backbone for car detection.

[66]
```python
class CarDetector:
    def __init__(self):
        self.model = fasterrcnn_resnet50_fpn(pretrained=True)
        self.model.eval()

    def detect_cars(self, image_path):
        image = Image.open(image_path)
        image_tensor = F.to_tensor(image)
        outputs = self.model([image_tensor])

        bounding_boxes = []
        class_0_count = 0  # Counter for class 0
        for box in outputs[0]['boxes']:
            x_min, y_min, x_max, y_max = box.tolist()
            class_label = 0  # Assuming class label 0 for cars
            if class_label == 0:
                class_0_count += 1
            x_center = (x_min + x_max) / 2
            y_center = (y_min + y_max) / 2
            width = x_max - x_min
            height = y_max - y_min
            bounding_boxes.append([class_label, x_center, y_center, width, height])

        return image, bounding_boxes, class_0_count
```

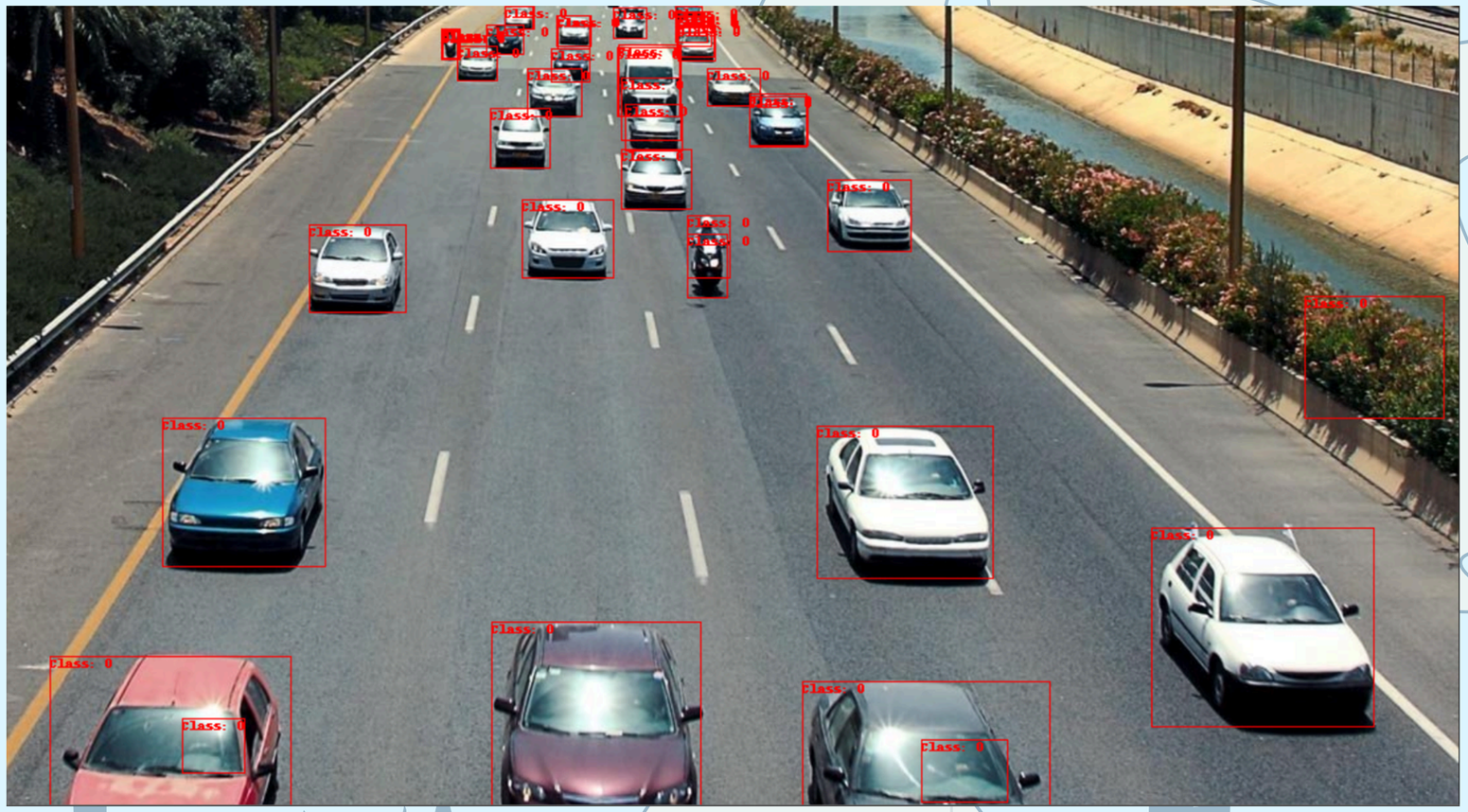
BEFORE DETECTION

RESULTS



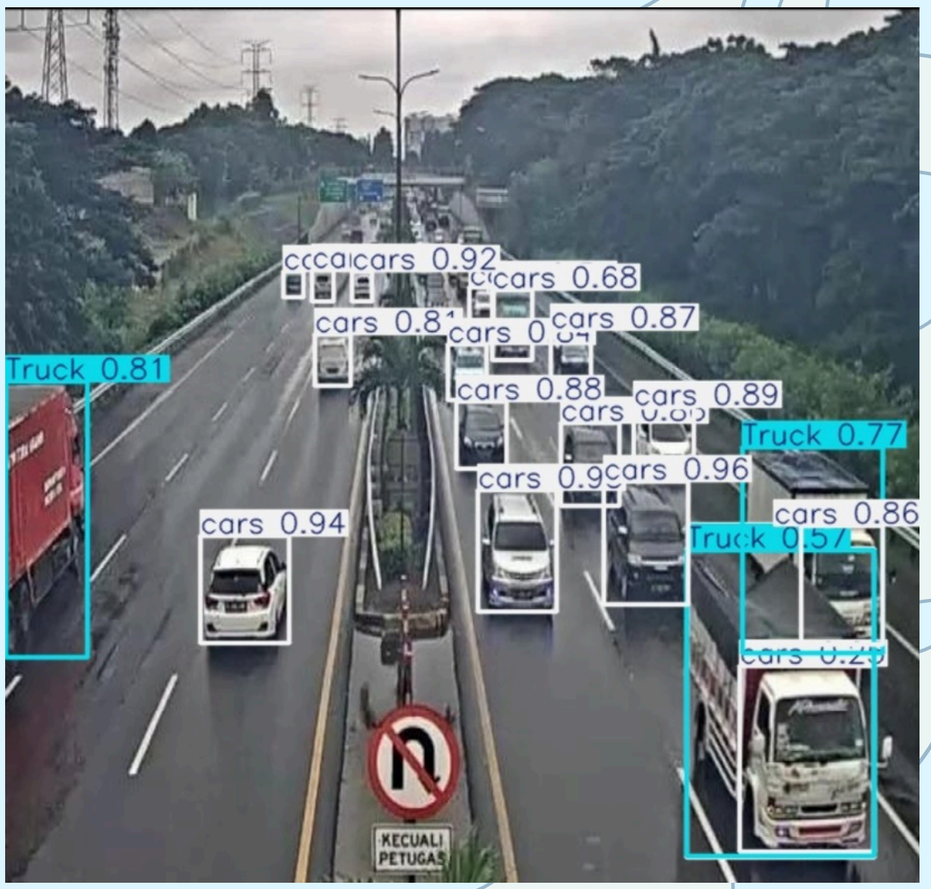
Number of cars detected: 17

RESULTS



```
Number of class 0 labels detected: 44
```

RESULTS




```
Model summary (fused): 168 layers, 3,006,233 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:14<00:00,  1.61s/it]
                   all        280       3292      0.637       0.67      0.671      0.579
                   Bus         59         64      0.599      0.211      0.302      0.266
                 Truck        231        585      0.581      0.868      0.785      0.671
                  cars        280       2643      0.731       0.93      0.926      0.799
```

- **What challenges did you face during data collection and how did you overcome them?** Data Availability and Quality.

- **Why did you choose your specific model architecture?** When we suggested the CNN architecture, we was drawing on common patterns and best practices that are prevalent in the field of computer vision.

- **How does the performance of your chosen model compare to alternative approaches, and what factors contributed to the differences?** The ability to learn complex features from large datasets and adapt to variations.

- **If you were to extend this project, what additional features or improvements would you consider?** Enhanced Object Recognition , Real-Time Applications ,Connect it withe the traffic light

BONUS TASKS

	AccX	AccY	AccZ	GyroX	GyroY	GyroZ	Class	Timestamp
0	0.758194	-0.217791	0.457263	0.000000	0.000000	0.000000	AGGRESSIVE	818922
1	0.667560	-0.038610	0.231416	-0.054367	-0.007712	0.225257	AGGRESSIVE	818923
2	2.724449	-7.584121	2.390926	0.023824	0.013668	-0.038026	AGGRESSIVE	818923
3	2.330950	-7.621754	2.529024	0.056810	-0.180587	-0.052076	AGGRESSIVE	818924
4	2.847215	-6.755621	2.224640	-0.031765	-0.035201	0.035277	AGGRESSIVE	818924
...	...	...	...	...	...	...	...	...
6723	0.915688	-2.017489	1.687505	0.450360	0.384845	-1.236468	SLOW	3583789
6724	-1.934203	0.914925	-0.096013	0.321468	0.649350	-0.477162	SLOW	3583790
6725	-0.222845	0.747304	-0.887430	0.361174	-0.406836	0.054291	SLOW	3583790
6726	-0.349423	0.067261	0.394368	-0.132405	0.020159	-0.004963	SLOW	3583791
6727	-0.402428	0.406218	-0.423009	-0.053603	-0.006720	0.001145	SLOW	3583791

6728 rows × 8 columns

BONUS TASKS

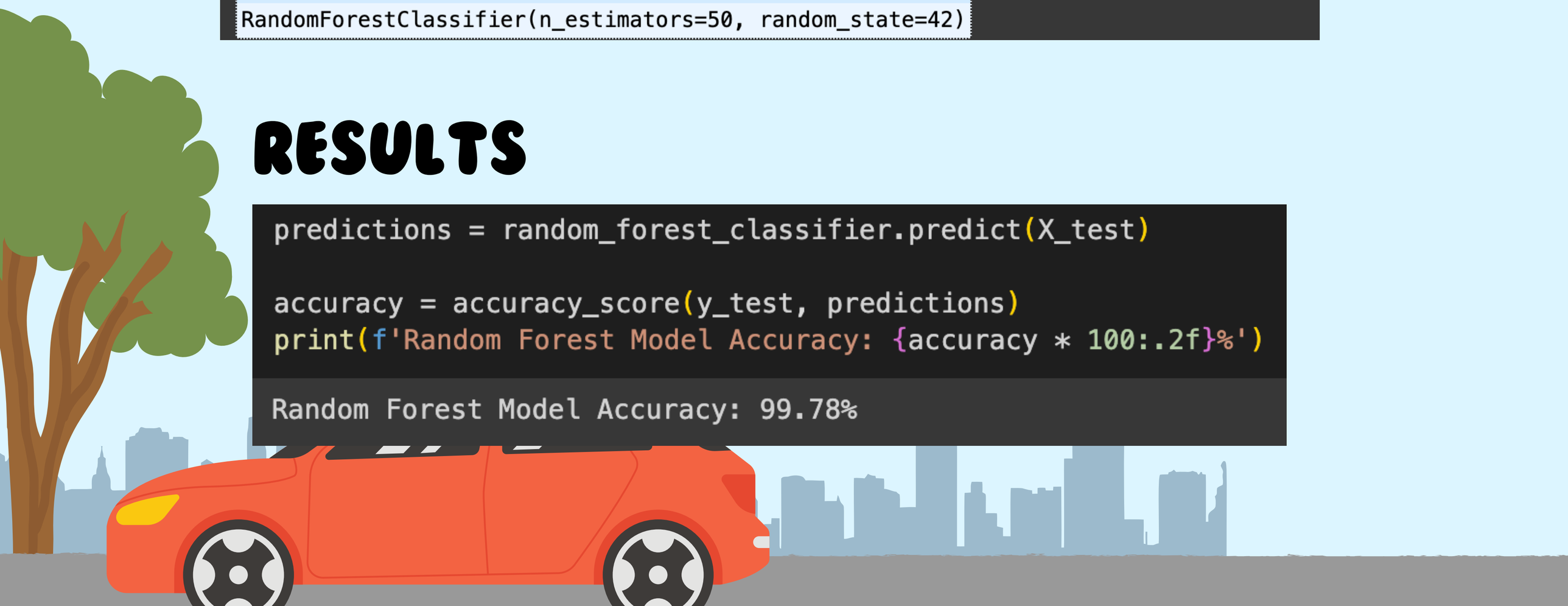
```
random_forest_classifier = RandomForestClassifier(n_estimators=50, random_state=42)
random_forest_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=50, random_state=42)
```

RESULTS

```
predictions = random_forest_classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f'Random Forest Model Accuracy: {accuracy * 100:.2f}%')
```

```
Random Forest Model Accuracy: 99.78%
```



CONCLUSION

Our project has the potential to significantly improve traffic efficiency and reduce congestion, leading to:

- **Shorter Travel Times**: Drivers will experience reduced wait times at intersections, saving time and fuel.
- **Reduced Emissions**: Optimized traffic flow will minimize idling time and fuel consumption, contributing to a cleaner environment.
- **Improved Safety**: Smoother traffic flow can lead to fewer accidents and a safer driving experience.

TEAM MEMBERS

- ALYAA
- KHUZAMA
- SALEH
- ANAS

THANK YOU