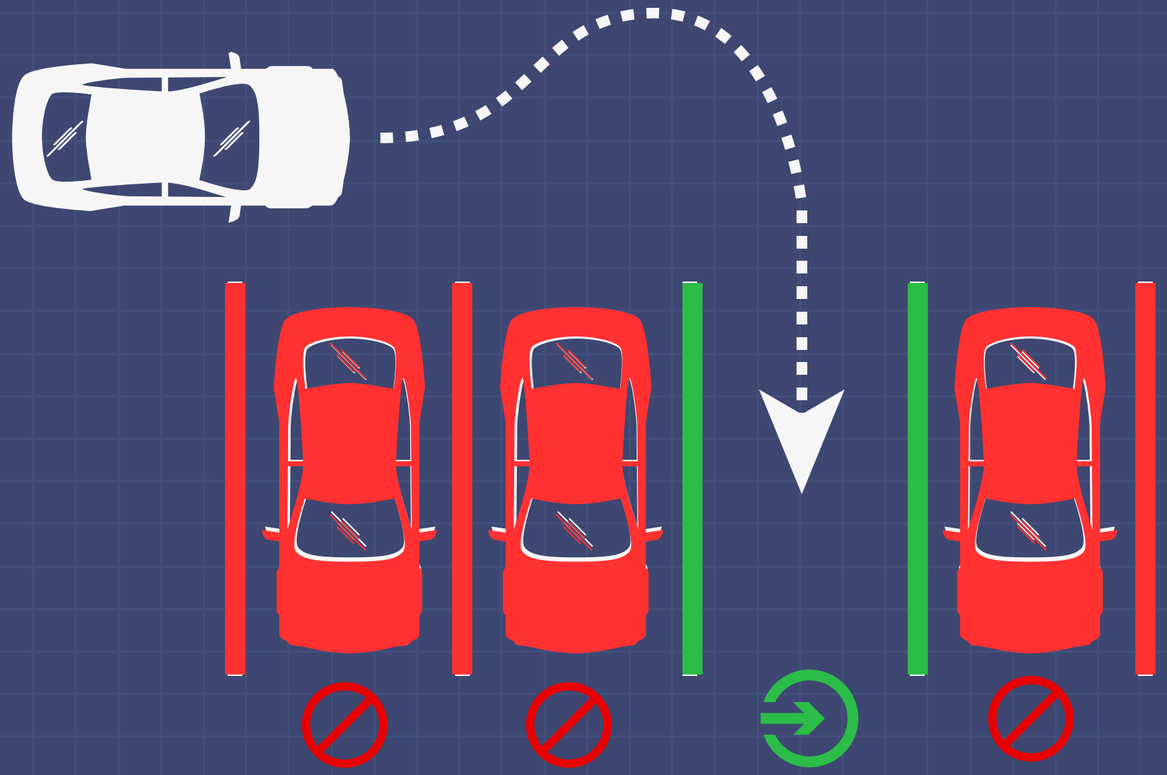


Deep Learning Project

Real-time Prediction of Parking space Availability

Presented by:

BENABOUD Oumaïma

MHASNI Khalid

Problem Statement

- Drivers spend **17 hours** a year looking for a Parking space.
- Parking search traffic creates **1.3 kg of CO_2** emissions per search.
- The fact that the average person spends almost an **entire day** of the year simply looking for parking space, whilst generating so many harmful emissions, proves that municipalities and parking operators need to invest in **smart parking technology**.

Available Solutions

"Several innovative solutions offer promising avenues to tackle the issue of parking inefficiency and its associated environmental impact."

Counter based Systems

- *Inability to track individual spaces*
- *Limited functionality: Relies solely on car counts*

Wired and wireless sensor-based parking

- *Installation costs*
- *Maintenance requirements*
- *Potential inaccuracies*

Camera-based systems

- *No additional cost : use existing CCTV networks*
- *Integration of deep learning*

Our Solution : Camera-Based System + Deep Learning integration

Let’s answer these Question to Validate the need of our solution

What? Integrating Camera-Based Systems with Deep Learning for real-time parking availability.

Collaboration between parking operators, municipalities. **Who?**

Where? Deployment in urban, suburban, commercial, and public parking areas.

Implementation post-infrastructure setup for continual model enhancement. **When?**

WHY? Cost-effective precision through Deep Learning models.

Training a model on parking lot images for accurate space identification and setup camera so it can work properly. **How?**

Our Process of making the classification model

Create a custom dataset with labelled images using Roboflow

Preprocess the data and use augmentation methods with Roboflow

Train the model using the YOLOv8s command line utility (Transfer Learning)

Validate the trained model's performance and perform inference on a test dataset to detect available parking spaces

Ensure the real-time feature in detecting parking space availability

Some unfamiliar terms :

What is Yolov8?

- **YOLOv8** is the latest version of the YOLO family of models, developed by Ultralytics.
- **YOLO** stands for **You Only Look Once**, and this series of models is named for its ability to predict every object present in an image.
- **YOLO** models are pre-trained on extensive datasets such as COCO and ImageNet.
- They provide highly accurate predictions for the classes on which they are pre-trained and can also learn new classes relatively easily.
- **YOLOv8** is designed for tasks such as object detection, classification, and segmentation.

Some unfamiliar terms :

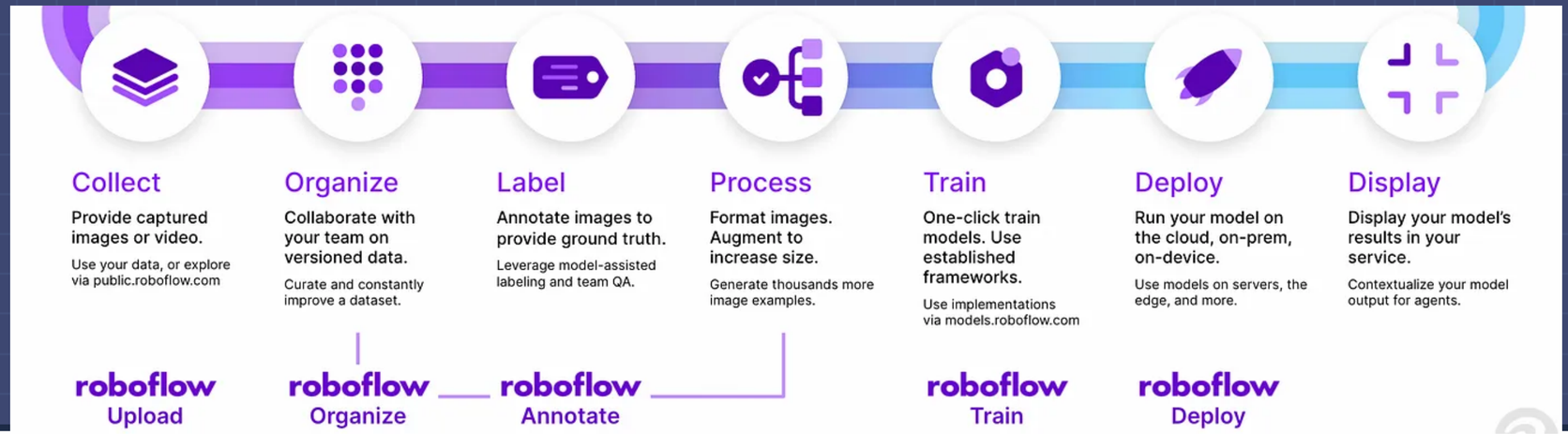
How does it work?

- **Transfer Learning with YOLOv8**: YOLOv8 utilizes pre-trained weights from datasets like COCO for general object recognition.
- **Adaptation to Custom Dataset:** The model fine-tunes its knowledge through transfer learning, aligning with the specific objects in our dataset.
- **Training Stage:** The model analyzes annotated images, learning to predict bounding boxes and class probabilities.
- **Evaluation and Continuous Improvement:** The model's performance is evaluated using a separate validation and test dataset, assessing metrics like precision, and recall.

Some unfamiliar terms :

What is Roboflow?

- **Roboflow**: An all-in-one platform simplifying **dataset management**, annotation, and preparation for machine learning tasks.
- Enables easy dataset upload, preprocessing, and augmentation for efficient model training in computer vision applications.
- **Features** include :



Project development process :

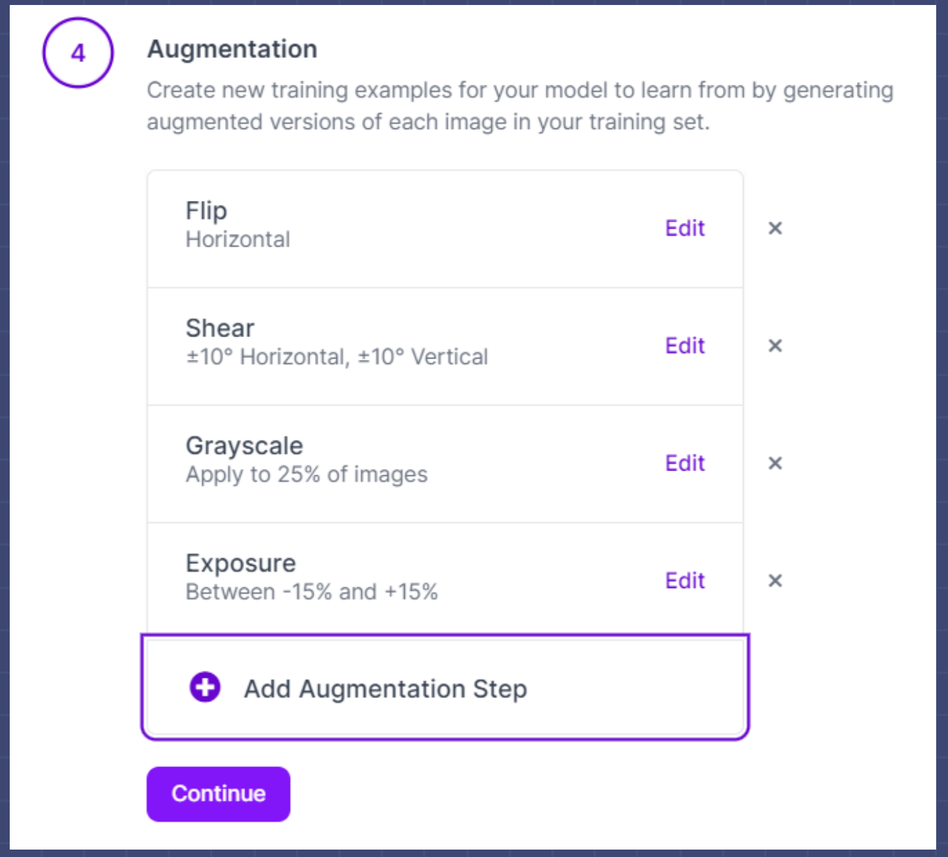
Labeling and annotating the Data using Roboflow

- **Upload** training video data to **Roboflow** Annotate, containing parking lot footage.
- **Divide** the video into **2 frames per second** for smoother annotation.
- **Open** and **annotate** each frame individually, marking occupied and empty parking spaces with bounding boxes.
- **Save** the annotated data.

Project development process :

Use augmentation methods with Roboflow

- Preprocess the data, integrating augmentation techniques:
  - Flip: Horizontal
  - Shear: ±10° Horizontal, ±10° Vertical
  - Grayscale: Apply to 25% of images
  - Exposure: Adjust between -15% and +15%
- Export the annotated data in a format compatible with YOLOv8.

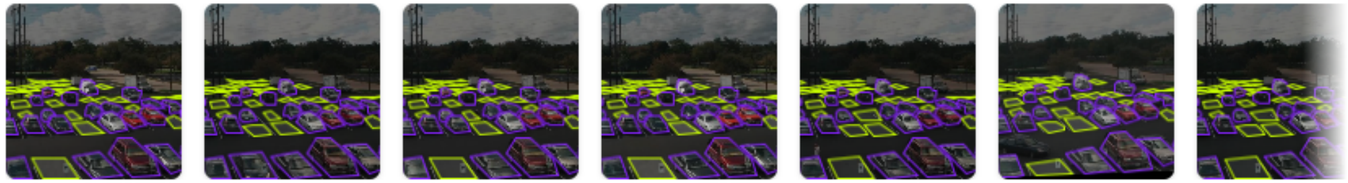


Project development process :

What does our Dataset look like after the preprocessing phase ?

720 Total Images

View All Images →



Dataset Split

TRAIN SET	VALID SET	TEST SET
88%	8%	4%
630 Images	60 Images	30 Images

Preprocessing

Auto-Orient: Applied

Resize: Fit within 640x640

Augmentations

Outputs per training example: 3

Flip: Horizontal

Shear: ±10° Horizontal, ±10° Vertical

Grayscale: Apply to 25% of images

Exposure: Between -15% and +15%

Project development process :

Uploading the custom dataset using Roboflow api


```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="8KftL2XAUUepBrqHNjMJ")
project = rf.workspace("mi3ad").project("parkingspace-neflg")
dataset = project.version(2).download("yolov8")
```

```
Collecting roboflow
  Downloading roboflow-1.1.12-py3-none-any.whl (68 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 68.5/68.5 kB 1.2 MB/s eta 0:00:00
Collecting certifi==2023.7.22 (from roboflow)
  Downloading certifi-2023.7.22-py3-none-any.whl (158 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 158.3/158.3 kB 3.4 MB/s eta 0:00:00
Collecting chardet==4.0.0 (from roboflow)
  Downloading chardet-4.0.0-py2.py3-none-any.whl (178 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 178.7/178.7 kB 11.6 MB/s eta 0:00:00
Collecting cycler==0.10.0 (from roboflow)
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting idna==2.10 (from roboflow)
  Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 kB 7.8 MB/s eta 0:00:00
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.23.5)
Collecting opencv-python-headless==4.8.0.74 (from roboflow)
  Downloading opencv_python_headless-4.8.0.74-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (49.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.1/49.1 MB 20.9 MB/s eta 0:00:00
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (9.4.0)
Collecting pyparsing==2.4.7 (from roboflow)
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
```

Project development process :

Train the model using YOLOv8s (Transfer Learning)

- We trained the model YOLOv8 on our custom dataset in 50 epochs.
- Training Details: By default, the SGD optimizer was used with a learning rate of 0.01.
- We also used the GPU of Colab (Tesla T4) that 16Gb in Vram

```
!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True
```

```
Overriding model.yaml nc=80 with nc=2

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.Conv                  [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.Conv                  [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.C2f                   [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.Conv                  [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.C2f                   [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.Conv                  [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.C2f                   [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.Conv                  [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.C2f                   [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.SPPF                  [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.Concat                [1]
 12                  -1  1    591360  ultralytics.nn.modules.C2f                   [768, 256, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.Concat                [1]
 15                  -1  1    148224  ultralytics.nn.modules.C2f                   [384, 128, 1]
 16                  -1  1    147712  ultralytics.nn.modules.Conv                  [128, 128, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.Concat                [1]
 18                  -1  1    493056  ultralytics.nn.modules.C2f                   [384, 256, 1]
 19                  -1  1    590336  ultralytics.nn.modules.Conv                  [256, 256, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.Concat                [1]
 21                  -1  1   1969152  ultralytics.nn.modules.C2f                   [768, 512, 1]
 22        [15, 18, 21]  1   2116822  ultralytics.nn.modules.Detect                [2, [128, 256, 512]]
Model summary: 225 layers, 11136374 parameters, 11136358 gradients, 28.6 GFLOPs

Transferred 349/355 items from pretrained weights
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.001), 63 bias
```

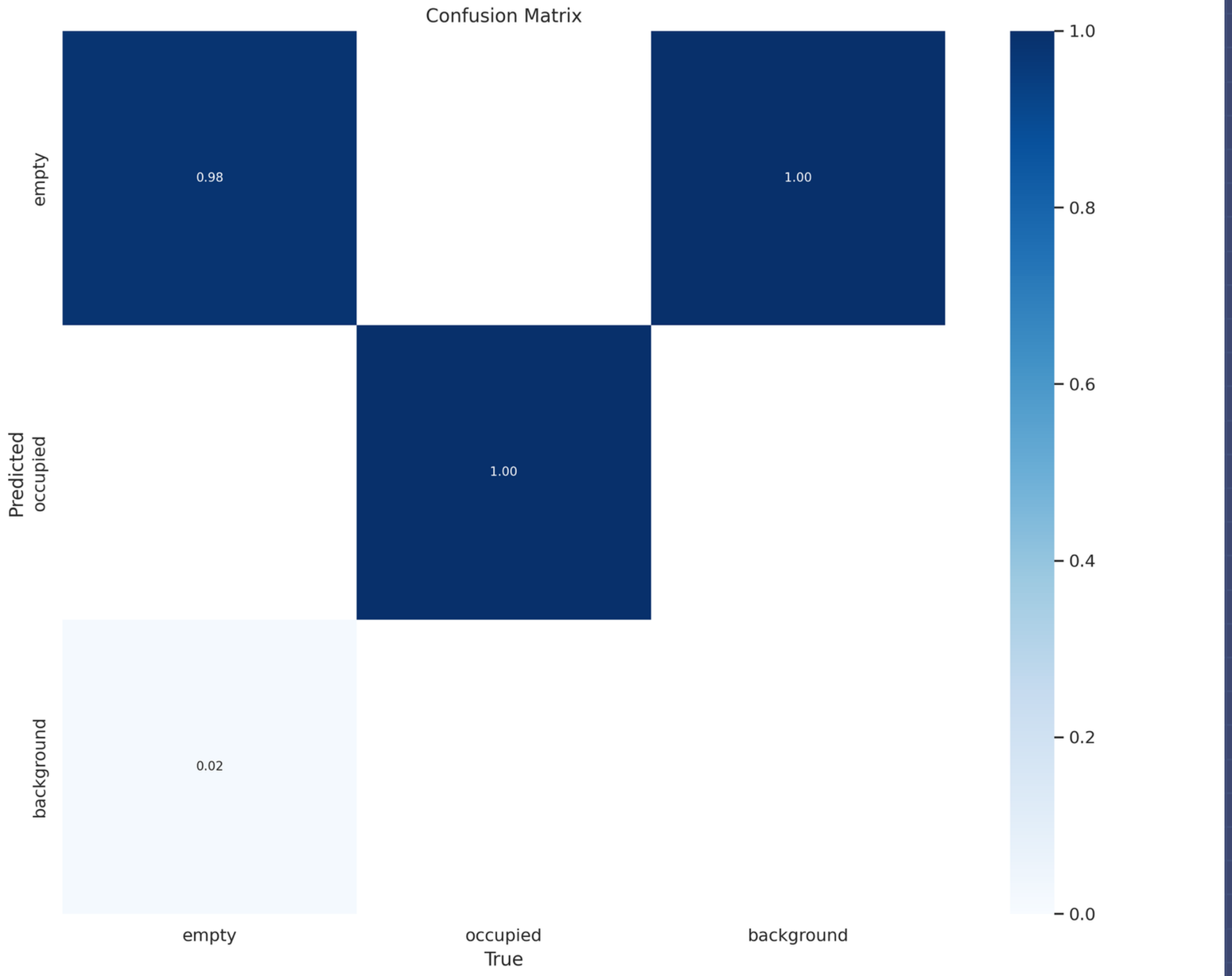
Project development process :

Train the model using YOLOv8s (Transfer Learning)

```
50 epochs completed in 1.968 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11126358 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 2/2 [00:08<00:00,  4.02s/it]
                   all         60       5160      0.998      0.988      0.991      0.934
                 empty         60       3194      0.997      0.977      0.988      0.884
              occupied         60       1966          1      0.999      0.995      0.985
Speed: 0.1ms pre-process, 3.7ms inference, 0.0ms loss, 2.3ms post-process per image
```

- Training Details: It took about 1.968 hours = 1h58mn.
- Performance Metrics: The model's performance was evaluated after each epoch. It reports metrics like Precision (P), Recall (R), and mean Average Precision (mAP) at different IoU thresholds (mAP50, mAP50-95) for each class ('empty' and 'occupied') and combined ('all').
- IoU (Intersection over Union): is a measure used to determine the overlap between the predicted bounding box and the ground truth bounding box of an object. It's calculated as the ratio between the intersection area and the union area of the two bounding boxes. If the IoU value exceeds a certain threshold (often 0.5 or 0.75), the detection is considered a true positive; otherwise, it's a false positive.


Confusion Matrix
empty
Predicted
occupied
background
0.98
1.00
1.00
0.02
empty
occupied
background
True
1.0
0.8
0.6
0.4
0.2
0.0

Project development process :

Validate the trained model's performance

```
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml

/content
2023-12-22 19:58:25.796368: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory:
2023-12-22 19:58:25.796424: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory:
2023-12-22 19:58:25.797826: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factor
2023-12-22 19:58:26.849816: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11126358 parameters, 0 gradients, 28.4 GFLOPs
val: Scanning /content/ParkingSpace-3/valid/labels.cache... 60 images, 0 backgrounds, 0 corrupt: 100% 60/60 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):  75% 3/4 [00:11<00:04,  4.42s/it]WA
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 4/4 [00:18<00:00,  4.53s/it]
                   all         60       5160      0.998      0.972      0.982      0.929
                 empty         60       3194      0.997      0.961      0.974      0.879
              occupied         60       1966          1      0.983      0.989      0.979
Speed: 3.2ms pre-process, 34.7ms inference, 0.0ms loss, 34.8ms post-process per image
```

- The model used for validation is the best.pt file
- The validation dataset contains 60 images.
- The model evaluated these images and made predictions on them, reporting several metrics for different classes ('empty' and 'occupied') and the overall metrics ('all').

Project development process :

Perform inference on the test dataset

```
!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.70 source={dataset.location}/test/images save=True line_thickness=1 iou=0.85
```

```
import glob
from IPython.display import Image, display

for image_path in glob.glob(f'{HOME}/runs/detect/predict/*.jpg')[:3]:
      display(Image(filename=image_path, width=640))
      print("\n")
```

Project development process :

Ensure the real-time feature in detecting parking space availability

Future work

- **Image data in diverse weather conditions:** Expanding the dataset to include images captured in various weather conditions, such as rain, snow, or fog, to enhance the model's adaptability to different environmental scenarios.

- **Night lighting considerations:** Incorporating image data captured during nighttime or low-light conditions to enable the model to accurately detect parking space occupancy in challenging lighting situations.

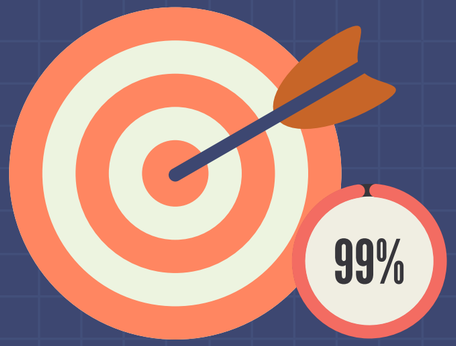
- **Addressing ambiguity through segmentation:** Implementing segmentation methods to better handle scenarios where parking spaces are partially occupied.

Conclusion

- The model showcased an impressive precision of **99.7%** on the validation dataset, indicating its strong potential as an affordable and dependable solution for outdoor smart parking technology systems.

- However, despite these achievements, several **challenges** hinder its performance. These challenges include detecting parking spaces during nighttime, handling shadows cast by buildings on the parking areas, and accurately identifying vehicles parked outside or between designated bays.

- Addressing these challenges is crucial for enhancing the model's accuracy and ensuring its effectiveness across **diverse conditions**.



Deep Learning Project

Thank you for your attention