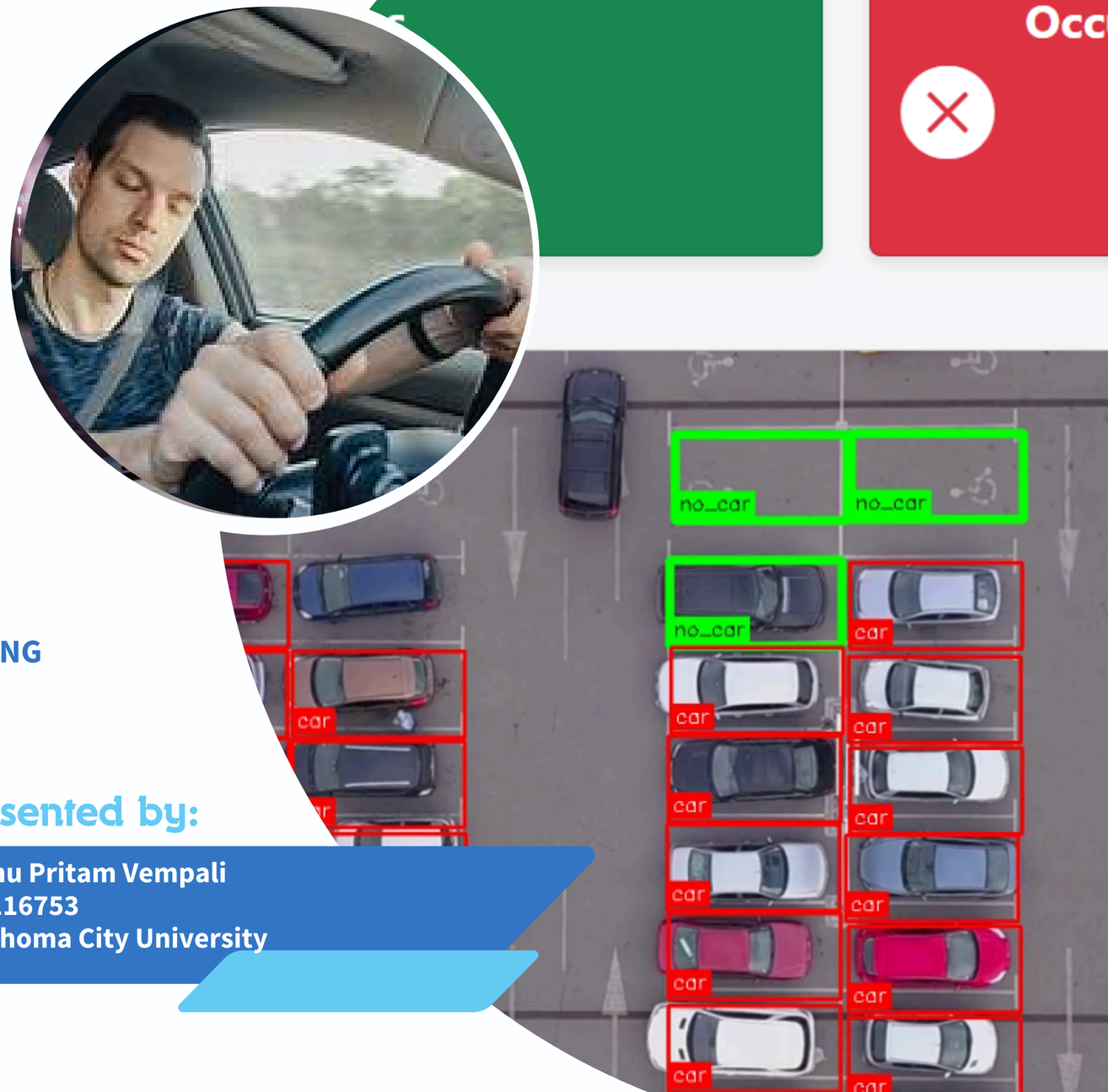
FINAL PRESENTATION

PARKING SPACE DETECTION USING TRANSFER LEARNING

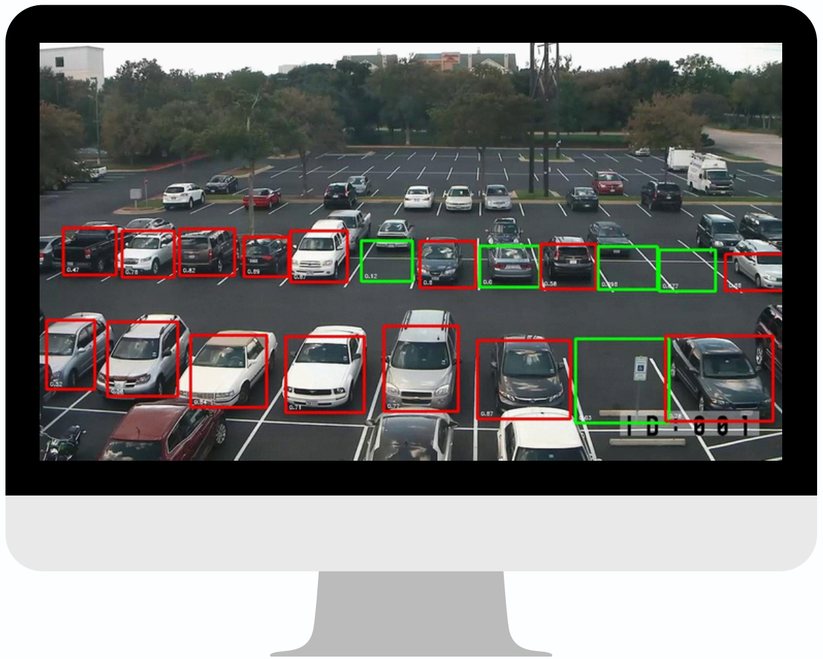


Presented by:

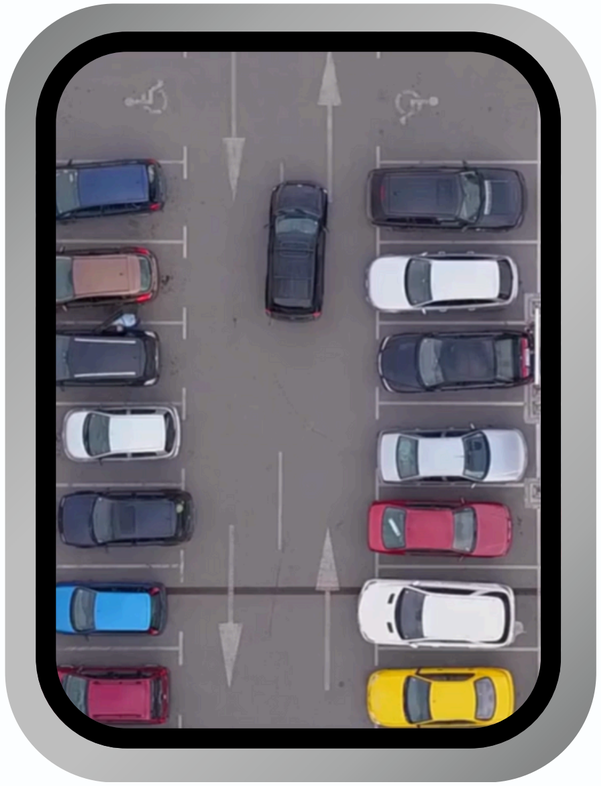
Bhanu Pritam Vempali
b00116753
Oklahoma City University

OVERVIEW

1. Mission
2. Data Collection & Pre-Processing
3. Model Training & Performance
4. Limitations of Classification Approach
5. Faster-RCNN Integration
6. Iou Implementation in Our System
7. Current Status & Debugging
8. Packages & Libraries



MISSION



Developed a real-time parking space detection system in two phases:

- Phase 1: Used a deep learning model (VGG16) to classify individual parking spots as "car" or "no_car" using manually cropped images, achieving high accuracy.
- Phase 2: Upgraded to a more advanced model (Faster R-CNN) that detects vehicles directly from full video frames using the PKLot dataset. This made the system more scalable and fully automated for real-world use.

Later slides explain how we determine whether a parking spot is occupied using detected vehicle positions.

DATA COLLECTION & PRE-PROCESSING

DATA COLLECTION

Regions of Interest (ROIs) were selected using an interactive OpenCV tool — left-clicking to select parking spots and right-clicking to undo. Each ROI was cropped to 130×65 pixels and saved in the cropped_img/ directory. Their coordinates were stored in carposition.pkl. The dataset was then organized into training (432 images) and validation (164 images) sets, with separate subfolders for each class.

PRE-PROCESSING

All images were standardized to 48×48 pixels to maintain consistent input dimensions. The dataset was split into 72.5% training and 27.5% validation images. Data augmentation was applied using ImageDataGenerator, including random rotations, shifts, flips, brightness adjustments, and normalization, improving model generalization.

MODEL TRAINING & PERFORMANCE

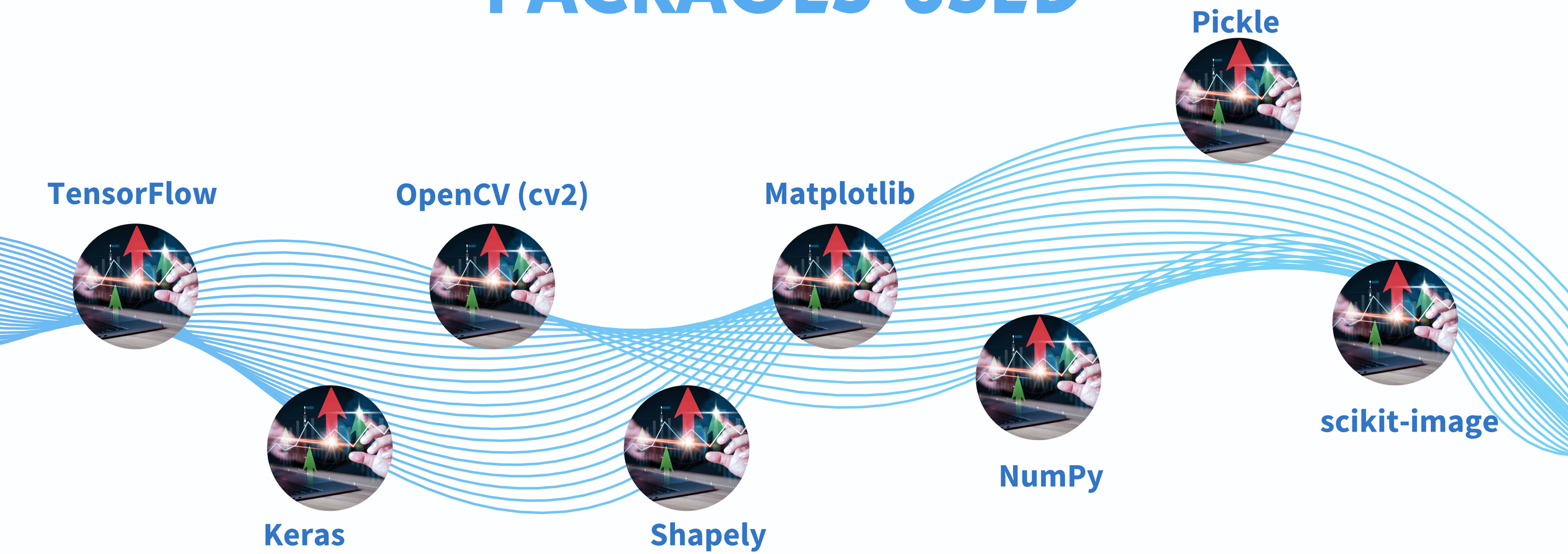
MODEL TRAINING

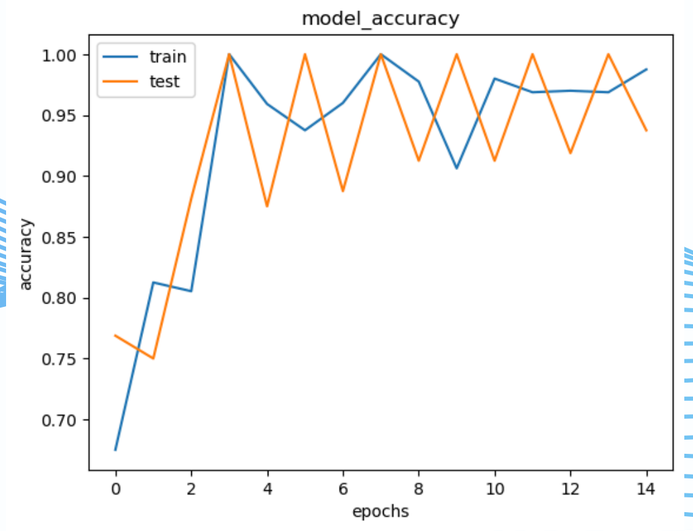
Transfer learning was implemented using the VGG16 model pretrained on ImageNet. The top layers were removed, and a custom classifier head was added. The first 10 layers were frozen to retain base features. The network was trained on 48×48 input images using a batch size of 32 for 15 epochs.

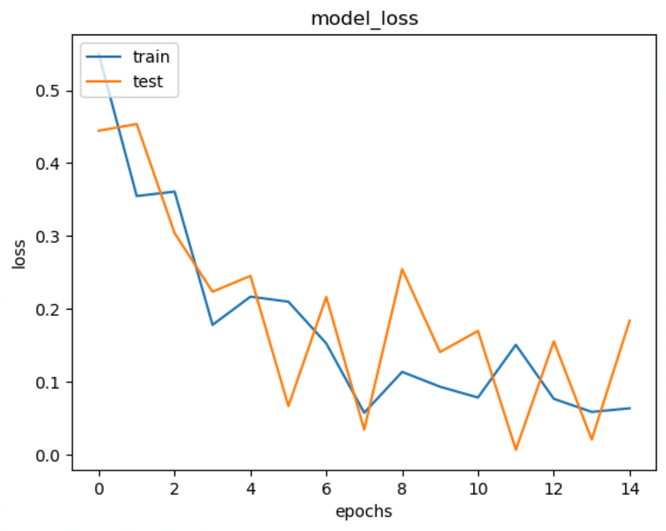
PERFORMANCE

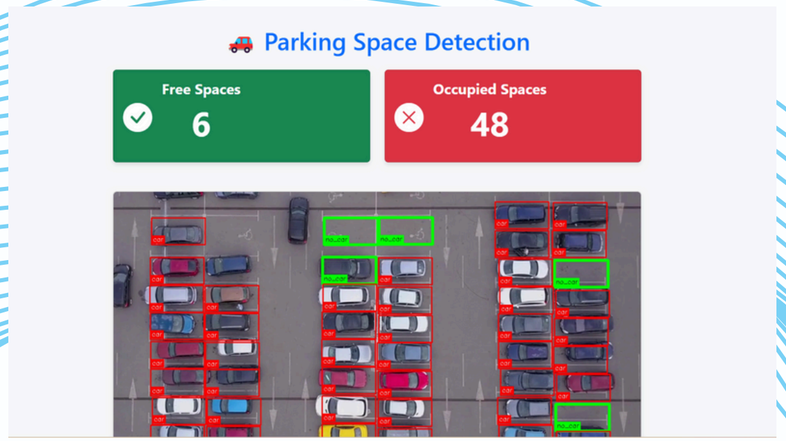
Training accuracy increased from 67.5% to 98.8%, with validation accuracy peaking at 100% and stabilizing at 93.8%. Training loss dropped from 0.55 to 0.06, while validation loss fluctuated but ended at 0.18. The model showed strong learning performance with effective generalization.

PACKAGES USED




model_accuracy
train
test
accuracy
epochs
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
2
4
6
8
10
12
14


model_loss
train
test
loss
epochs
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
10
12
14


Parking Space Detection
Free Spaces
6
Occupied Spaces
48

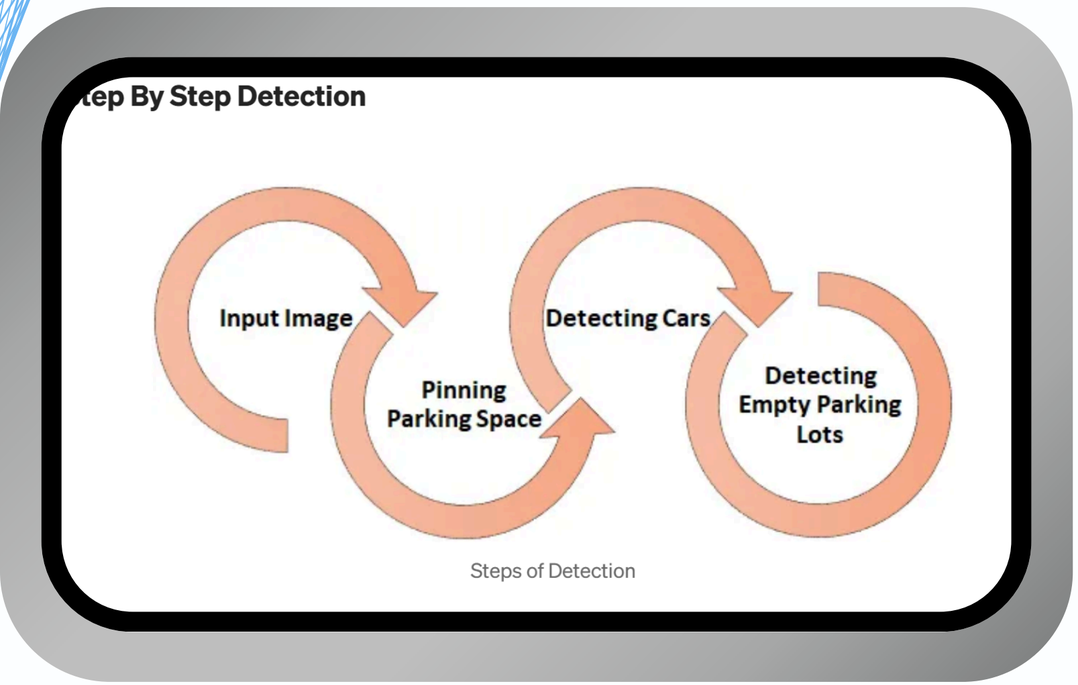
LIMITATIONS OF TRANSFER LEARNING

Manual ROI Selection:
Parking space regions must be manually annotated and cropped in advance, limiting automation and scalability.

Sensitivity to Angle/Lighting:
Misclassifications can occur due to changes in camera angle, shadows, or low lighting conditions.

Static Coordinates:
Detection relies on fixed parking spot positions, making it ineffective for dynamic or unstructured parking lots.

FASTER R-CNN INTEGRATION



To overcome the limitations of manual ROI-based classification:

- We integrated Faster R-CNN with ResNet-50 FPN backbone from PyTorch's torchvision.models.detection.
- Modified the final layers to support custom parking class detection.
- Trained the model using a COCO-style formatted version of the PKLot dataset.
- Faster R-CNN outputs bounding boxes and class probabilities, used to check overlap with fixed parking slots.

✓ Compatible with current PyTorch setup

✓ No TensorFlow/Keras issues

✓ Faster to integrate, supports batch training, and provides confidence scores for detections

DATA COLLECTION & PRE-PROCESSING

DATA COLLECTION

We used the PKLot dataset, which already comes organized into training, validation, and test folders. It contains 12,416 images of real-world parking lots under different weather conditions (sunny, cloudy, rainy), with each parking space labeled as occupied or empty.

Annotations:

- The dataset originally labeled parking spaces using rotated rectangles.
- We converted these into standard bounding box format (top-left and bottom-right corners).
- The annotations were saved in COCO-style JSON files — a format widely used for object detection tasks.

PRE-PROCESSING

- We applied data augmentation to improve model generalization and handle real-world variations.
- Affine transformations (e.g., small shifts, scaling, and rotations)
- Color and brightness adjustments
- Normalization to standardize pixel values
- These steps help the model learn better, even if lighting or camera angles change.

MODEL TRAINING & PERFORMANCE

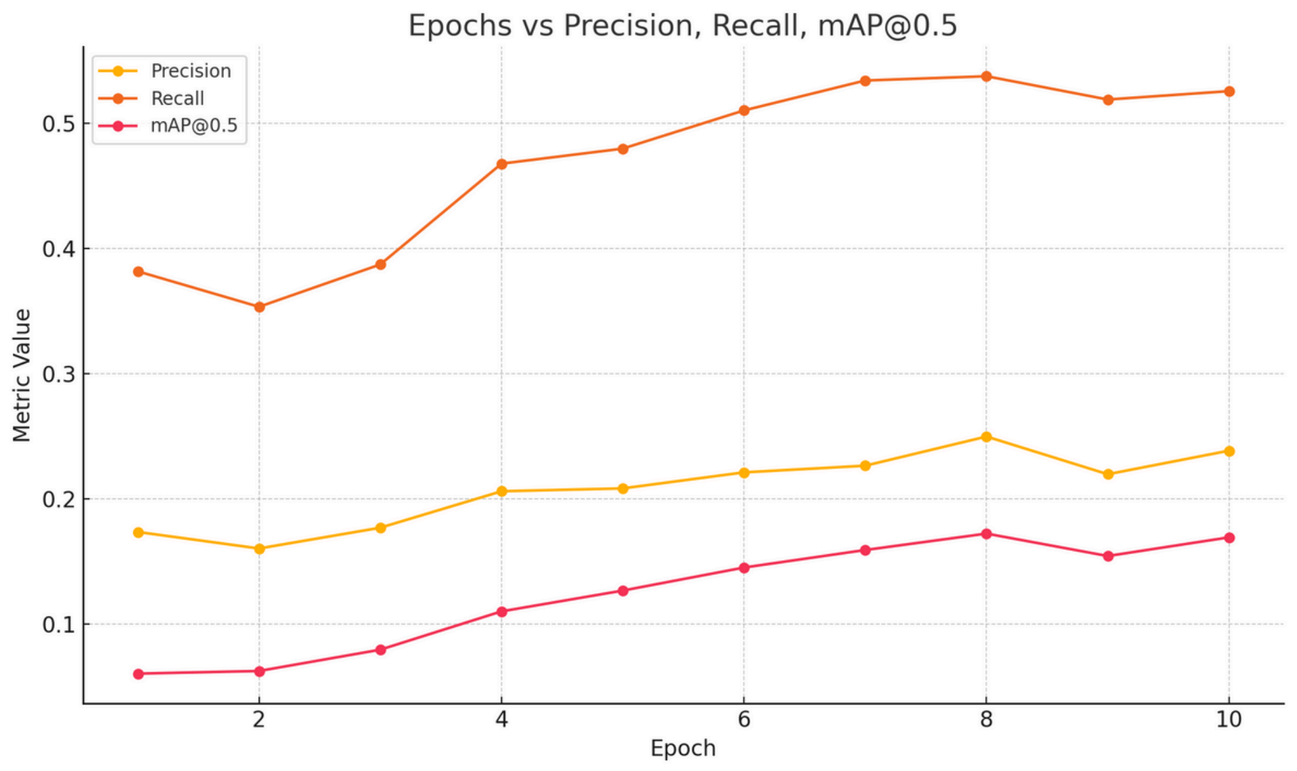
MODEL TRAINING

- We trained Faster R-CNN with a ResNet-50 FPN backbone for detecting vehicles from full-frame parking lot images.
- The model was fine-tuned using the PKLot dataset with bounding box annotations.
- Evaluation was done using mean Average Precision (mAP), Precision, and Recall metrics.

PERFORMANCE

- mAP@0.5: 0.1693
- mAP@0.5:0.95: 0.0855
- Precision: 0.2385
- Recall: 0.5256

after 10 epochs,The model is able to detect vehicles, but overall accuracy is still limited. This is due to less epoch training,varied lighting, partial occlusions, and class imbalance in the dataset.


Epochs vs Precision, Recall, mAP@0.5
Precision
Recall
mAP@0.5
Metric Value
0.1
0.2
0.3
0.4
0.5
2
4
6
8
10
Epoch


Free Spaces: 37/56

INTERSECTION OVER UNION (IOU)

IoU is a common method used in object detection to measure how well a predicted box overlaps with the ground truth (or any reference area like a parking slot).

- Imagine drawing a box around a detected vehicle and another box around the parking slot.
- IoU = Area of Overlap / Area of Combined Region

If a vehicle is only partially inside a slot or misaligned, IoU can be very low, even though the slot is clearly occupied.

WHY IOSA INSTEAD OF IOU?

In real-world parking detection, IoU isn't always reliable because:

- Vehicles don’t perfectly align with slot boxes.

- IoSA (Intersection over Slot Area) gives a more practical solution:
- It checks how much of the parking slot is covered by a vehicle.

- We used Shapely to compute the overlap between each detected box and the predefined slot polygons.
- Fits well with fixed-slot-based parking systems.

CONCLUSION

01

Faster R-CNN successfully replaced Mask R-CNN for detecting vehicles in parking lots, offering smooth integration using PyTorch without compatibility issues.

02

The system uses a more practical IoSA-based logic to determine parking space occupancy, improving robustness in real-world scenarios with varied alignment.

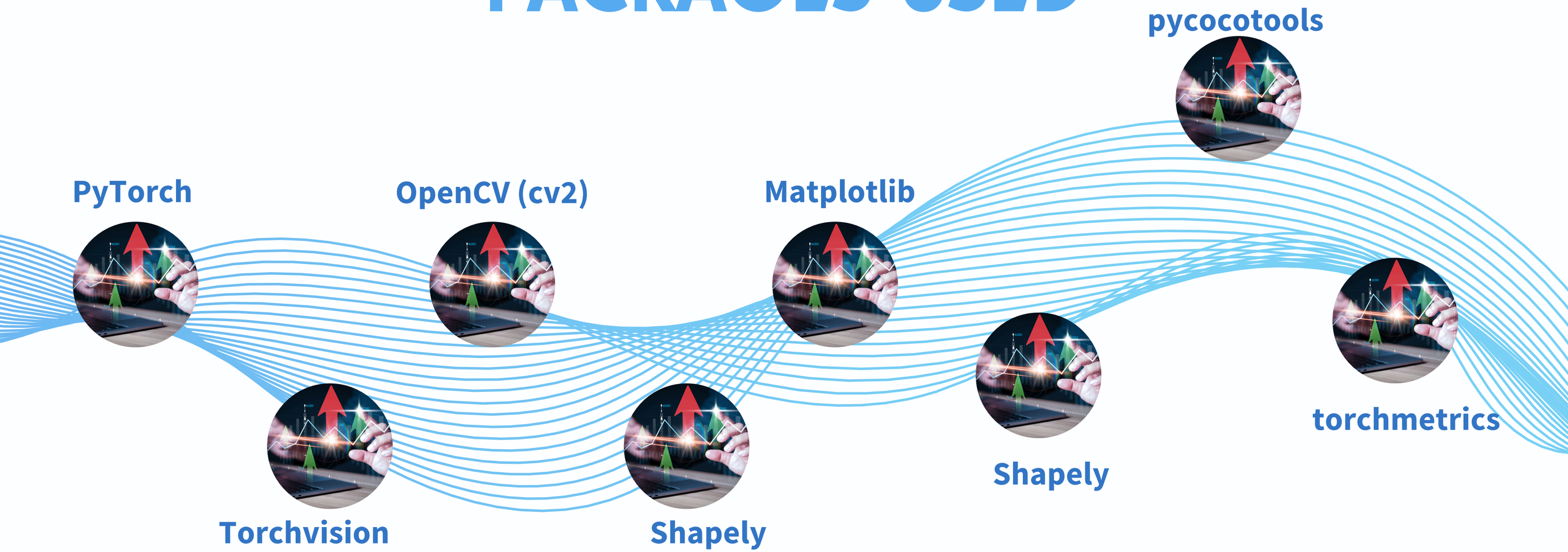
03

While the current model shows moderate accuracy (mAP@0.5 = 0.1693, Precision = 0.2385), it effectively detects vehicles and marks slot status in most cases.

04

Further training, better data balance, and using real-time optimized models (like YOLO) can significantly enhance both accuracy and inference speed.

PACKAGES USED



THANK YOU