SMART SEATS

Computer vision & Deep learning

MASTER DEGREE IN ARTIFICIAL INTELLIGENCE

University of Verona

2023/2024

Chiara Venturi (VR504017) & Dawood (VR512705)

Contents

1. INTRODUCTION

In recent years, there's been a growing need to use space better and improve safety in places like offices, auditoriums, and public transportation. Detecting if seats are occupied can help with these issues. Our project aims to create an Intelligent Transportation System, an AI-based system that can reliably monitor and manage seating in real-time. This fits into the areas of computer vision and artificial intelligence, especially for smart environments and public safety. [1]

The main problem we are solving is the lack of an accurate, automated way to check if seats are occupied in different settings. Traditional methods, like manual checks or pressure sensors, are often unreliable and take a lot of time. These issues can lead to wasted space, overcrowding, and safety problems, especially in public transport or emergency situations.

Our project aims to change how we monitor and manage seating. By using advanced AI, we can be more accurate and efficient than traditional methods, improving safety and convenience. This technology can also save money by reducing the need for manual checks and allowing for better maintenance of seating areas.

1.1. State of the art

Seat occupancy detection systems have become crucial in various applications, these systems aim to determine whether a seat is occupied, often using a combination of sensors, machine learning algorithms, and data analytics.

Traditional seat occupancy detection methods rely primarily on pressure sensors and mechanical switches, making them simple and cheap. These methods have some limits, they can lead to false positives due to object placement on seats, and they can also be easily broken.

With the rise of computer vision and deep learning, vision-based seat occupancy detection has gained significant traction. These methods use cameras to capture images or videos of the seating area, which are then processed using algorithms to detect occupancy, such as Convolutional Neural Networks.

1.2. Objectives

The primary objective of this project is to develop an accurate and reliable seat occupancy detection system using machine learning algorithms. This system aims to improve the safety, efficiency, and functionality of public transportation.

The increasing demand for public transportation has created the need to optimize resource utilization. Managing seat occupancy is crucial to ensuring a comfortable travel experience for passengers, especially during the COVID-19 pandemic when rules like personal distancing and avoiding overcrowded public spaces were in effect. Real-time information on available and occupied seats can offer several advantages: passengers can make more informed decisions about their travel, choosing less crowded seats. At the same time, companies can optimize passenger distribution across platforms, improving service efficiency and ensuring more effective use of transportation resources.

In our project, we will focus on public transportation, ensuring better distribution of passengers to reduce overcrowding and improve passenger comfort, allowing them to plan their journey according to their needs. [1]

2. METHODOLOGY

2.1. Methods and Algorithms

To achieve our project objectives, we will use a combination of object detection and pose estimation techniques. Specifically, we will use YOLO (You Only Look Once) for real-time object detection and compare it with Faster RCNN, and pose estimation algorithms to determine seat occupancy.

- **YOLO (You Only Look Once)**: This algorithm is chosen for its real-time detection capabilities. YOLO divides the image into a grid and predicts bounding boxes and probabilities for each grid cell, making it highly efficient for detecting seats in various environments. [5][6][8]
- **Faster RCNN**: this is chosen for its high accuracy in object detection for the seats. It uses a region proposal network to generate candidate object locations, which are then refined by a convolutional neural network. This model will be useful in scenarios where precision is critical. [7][8]
- **Pose Estimation Algorithms**: These algorithms will be used to determine if a person is sitting in a detected seat. Pose estimation will help differentiate between occupied and unoccupied seats by analyzing the human posture and key points. [5][6][8]

2.2. Dataset

We used the COCO (Common Objects in Context) dataset, a large-scale object detection, segmentation, and captioning dataset. This dataset is critical for training both the YOLO model for seat detection and the pose estimation algorithms, with its annotation we can fine-tune our models. COCO provides a diverse set of annotated images, which includes various object categories and human joints points, ensuring that our models learn to accurately identify seats and distinguish between occupied and unoccupied states.

We downloaded three image files, train2017, val2017, test2017 and one annotation file annotations_trainval2017. In the train folder we have 118k images, for validation we have 5k images. The test folder contains 40670 images. In the annotation folder we have person key points as well as instances for training and validation. [2][3][4]

2.3. Data Preparation

Pose Estimation Dataset

To train a YOLOv8 Pose estimation model the dataset should contain both images and their corresponding labels. We created .txt files for each image containing the necessary label information and saved them in the train and val folders. Additionally, we removed images that did not contain any persons, then we further reduced the images to 10k to support the training session in our own machine. [3]

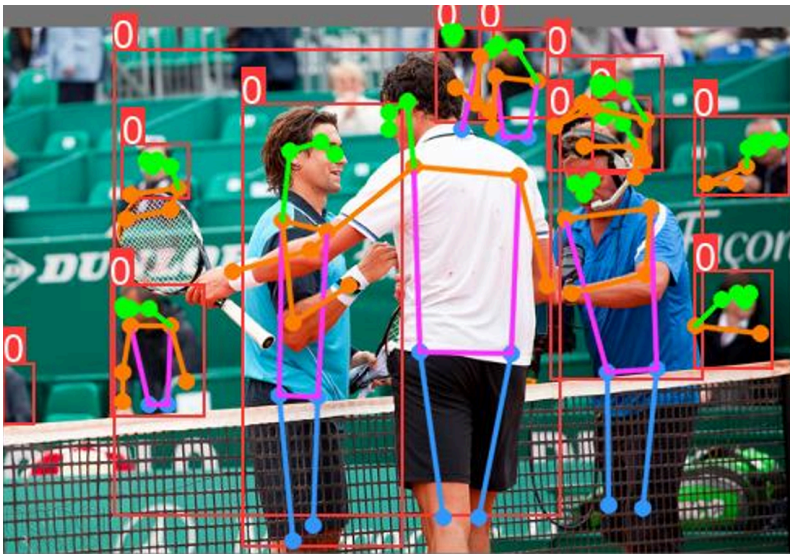

Figure 1: *sample image of a Person's key points from train batch*

FasterRCNN & Yolov8 (Detection Dataset)

To train a Faster R-CNN model, our dataset directory is structured to include separate folders for images and annotations. Within the images folder, we have subdivided the content into training and validation images. For this specific project, we focused exclusively on the 'chair' category, which is identified by ID 62 in the COCO dataset. We filtered out all other categories, retaining only the images that contain chairs from the "instances_train2017" and "instances_val2017" datasets. This process resulted in a training set with 12,774 chair images and a smaller validation set with 580 images.

We used the same image sets to train the YOLO model, which was also configured to detect only chairs. Additionally, for each image in our dataset, we generated a corresponding text file from the annotations, ensuring each image was paired with relevant annotation data for effective model training. [3][4]

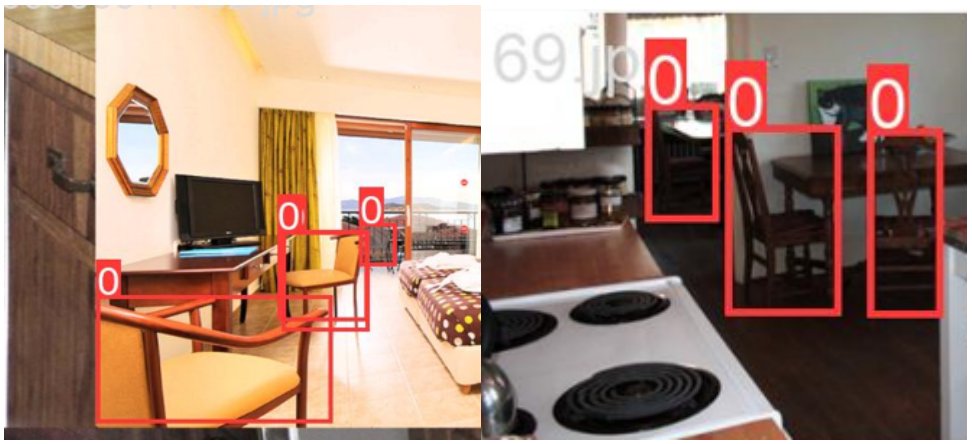

Figure 2: *Chairs labels from training batch*

2.4. Analytical and Computational Tools

To develop, train, and deploy our seat occupancy detection system, we will utilize a variety of analytical and computational tools:

TensorFlow and PyTorch: These deep learning frameworks will be used to develop and train our YOLO and pose estimation models. They provide the necessary tools and libraries for implementing neural networks. [7]

OpenCV: This open-source computer vision library will be used for image processing tasks, such as pre-processing input images and post-processing model outputs. [9]

NVIDIA CUDA and cuDNN: These technologies will accelerate our deep learning computations, allowing us to train models faster and more efficiently on GPUs. [10]

Jupyter Notebooks: For experimentation and iterative development, Jupyter Notebooks will be used to write and test code, visualize results, and document the development process.

2.5. Model training with YOLO

The training process included converting the COCO dataset to YOLO format, configuring the training parameters, and training the model.

We configured the YOLOv8 model by defining the data paths and setting up the training parameters. This included creating a data.yaml file that specifies the paths to the training and validation images, as well as the class names. We use the ultralytics library to train the model with 30 epochs and with image size of 640 pixels.

2.6. Model training with Faster RCNN

We trained the Faster R-CNN model using the Detectron2 library, developed by Facebook AI Research, which is a powerful tool for object detection tasks. The training process involved several steps, including dataset registration, model configuration, implementation of custom hooks for monitoring, and performance visualization.

We began by registering the COCO dataset for both training and validation, specifying the paths to the annotations and image directories. This ensured that our model had access to the necessary data for training and evaluation.

Next, we configured the Faster R-CNN model to detect a single class, "chair." The configuration was based on a pre-trained model from the Detectron2 model zoo, with weights initialized from a previously saved checkpoint. We set the confidence threshold for detection to 0.7 and adjusted the minimum input size for test images. Additionally, we specified training parameters such as batch size and learning rate to optimize the training process.

To further enhance the training, we implemented a custom hook that saved the model at regular intervals and printed the training losses, providing insights into the model's performance throughout the training process.

For model validation, we processed each frame of a video, performing chair detection on each one. We then stabilized the detected bounding boxes to improve accuracy, ensuring consistent and reliable detection results across the video.

2.7. Model training for pose estimation

To determine if a person is occupying a seat, we used pose estimation techniques that allows us to analyze human posture and key points, which is useful for identifying seated individuals. The training process involved preparing the dataset, converting annotations, configuring the training parameters, and training the pose estimation model.

We converted the COCO annotations to a format suitable for pose estimation training. To optimize the training process, we reduced the number of training images removing also the ones that did

not contain persons. This step ensured that we had a manageable dataset size without compromising the diversity of the data.
We used the YOLOv8 model (yolov8n-pose.pt) for training, using the ultralytics library.

2.8. Smart Seats model

The model script integrates object detection and pose estimation to classify human poses and detect seat occupancy in real-time. The system processes video frames using the YOLOv8 pose estimation model and the Faster R-CNN object detection one.
The YOLOv8 model is used for estimating human keypoints, which represent various body parts and their positions. Concurrently, the Faster R-CNN model detects chairs in each frame, identifying their bounding boxes.
To classify poses, the system extracts the key points from the pose estimation results and calculates angles between them. This involves analyzing the angles formed by the upper body (shoulders, hips, and knees) and the lower body (hips, knees, and ankles).
The algorithm first checks if all key points in the lower part of the body are visible: if they are, it calculates the angle at the knee, and if this angle is greater than 150 degrees, the pose is predicted as sitting; otherwise, it is standing. In some situations, key points on the lower parts of the body may not be visible, such as when the ankle keypoints are missing. In this case, the algorithm looks at the shoulders, hips, and knees and measures the angle at the hip: if this angle is greater than 150 degrees, the person is considered standing, otherwise is sitting.
For each detected chair, the system checks if the key points of the person fall within the chair's bounding box, if they indicate a sitting pose, the chair is marked as occupied. The algorithm also handles overlapping chair detections by considering the chair with the highest overlap, ensuring accurate occupancy detection.

3. EXPERIMENTS AND RESULTS

In this section, we talk about the experimental setup, evaluation protocols, conditions under which the tests were conducted, and the metrics used to measure the performance of our pose estimation and seat occupancy detection models.

3.1. Results for YOLOv8 Pose Estimation

In our experiments with the YOLOv8 pose estimation model, we focused on evaluating its ability to detect and classify human poses in a real-time environment.
The model was trained over 20 epochs, and these are the following key metrics.

Metric	**Value**
Average Box Loss	1.3492
Average Pox Loss	3.0355
Average KOBJ Loss	0.3471
Average CLS Loss	1.2080
Average DFL Loss	1.3031

3.1.1. Loss Functions

These parameters represent the weights assigned to different components of the overall loss function. Each type of loss addresses a specific aspect of the model's predictions, from how well it locates objects within a frame to how it identifies and classifies various key points:

- **box: 7.5 (Box regression loss)**
  This loss measures how accurately the bounding boxes around the detected objects are predicted. A higher weight for box regression loss suggests that accurate localization of objects is important for this task. An average box loss of **1.3492** indicates the model's performance in localizing the objects within the frames.
- **cls: 0.5 (Classification loss)**
  This loss assesses how well the model classifies the detected objects. A lower weight indicates that while classification is important, it might be less critical than predicting the bounding boxes and keypoints. In pose estimation, the primary goal is often to correctly identify the key points, so classification may not need to be that precise. An average classification loss of **1.2080** reflects the model's performance in identifying objects.
- **dfl: 1.5 (Distribution Focal Loss)**
  DFL is used to handle class imbalance by focusing more on hard-to-classify examples. An average DFL loss of **1.3031** shows how the model balances between easy and difficult examples.
- **pose: 12.0 (Pose estimation loss)**
  This loss evaluates the accuracy of the predicted key points against the ground truth ones. A high weight indicates that keypoint accuracy is of utmost importance for this model. it makes sense to prioritize this loss to ensure the model is highly accurate in predicting keypoints. An average pose loss of **3.0355** highlights the model's effectiveness in accurately predicting the keypoints.
- **kobj: 1.0 (Keypoint object loss)**
  This loss helps in associating the keypoints with the correct objects. So it helps the model understand which key points belong to which person in a scene, ensuring that the limbs and joints are correctly attributed and not mixed up between multiple people. An average KOBJ loss of **0.3471** shows how well the model differentiates between key points of different objects.

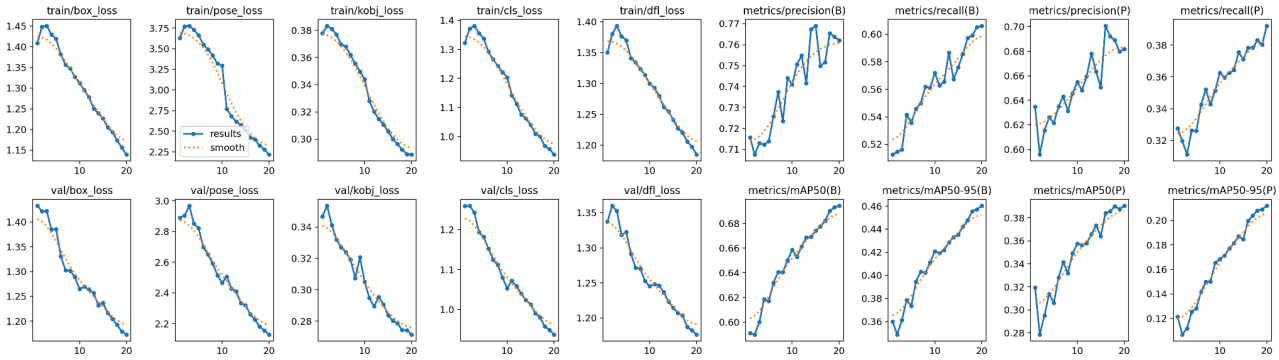

Figure 3: *Training and Validation Metrics for Pose Estimation*

The training and validation loss curves show a steady reduction in losses over 20 epochs. The metrics for precision, recall, and mAP steadily improve, reflecting enhanced model performance in pose estimation tasks.

3.1.2. Evaluation Metrics

The metrics provided represent the performance of the YOLOv8 pose estimation model, divided into two categories: B (bounding boxes) and P (pose keypoints). Each category is evaluated using precision, recall, and mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds.

Precision (B)	Recall(B)	mAP50 (B)	mAP50-95 (B)	Precision(P)	Recall (P)	mAP50 (P)	mAP50-95 (P)
0.762	0.606	0.694	0.46	0.681	0.392	0.39	0.212

Precision (B): Precision measures the accuracy of the detected bounding boxes A precision of 0.762 indicates that 76.2% of the bounding boxes predicted by the model were correct, which means a high level of accuracy in identifying the correct regions containing keypoints.
Recall (B): Recall measures tells us how good the model is at finding all the defects. A recall of 0.606 implies that the model identified 60.6% of all possible correct bounding boxes. This suggests that the model might miss some key points.
mAP50 (B): is an overall measure of how well the model performs. It combines both precision and recall into a single number. The value of 0.694 suggests that, on average, the model's predictions align well with the ground truth data when a 50% overlap is considered.
mAP50-95 (B): This metric is an average of the mAP calculated at different IoU thresholds, from 50% to 95%, in increments of 5%. A score of 0.46 indicates the model's performance across this range, it shows a decrease in accuracy as the IoU becomes stricter.

Precision (P): Similar to the bounding box precision, this measures the accuracy of the key points detected, with a score of 0.681 indicating that 68.1% of the detected key points were accurate.
Recall (P): The recall for keypoints is lower than the precision, at 0.392, suggesting that while the detected keypoints are accurate, the model fails to detect many key points across the dataset.
mAP50 (P): The mean Average Precision for keypoints at a 50% IoU threshold is at 0.39, indicating

moderate accuracy in aligning detected keypoints with their actual locations in the ground truth.
mAP50-95 (P): The average mAP from a 50% to 95% IoU threshold for keypoints shows a score of 0.212, which is quite low. This suggests that the model's performance significantly drops as the criteria for correct keypoint detection become more stringent.

3.1.3. Confusion matrix

The confusion matrix provides an insight of the classification capabilities of the model, it distinguishes between “person” and “background” classes. We have a high positive rate for the person class, confirming that the model identifies the right human figures. However we have some misclassifications that regard mainly complex backgrounds or overlapping subjects, leading to false negatives.

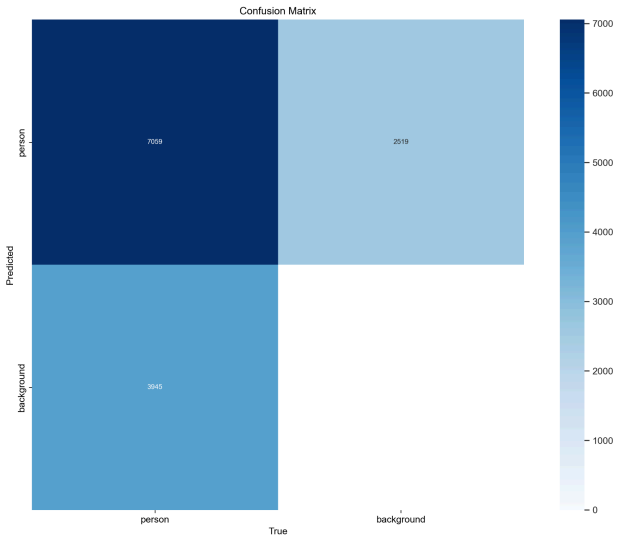

Figure 4: *Confusion Matrix Of Pose Estimation*

True Positives (7059): The model correctly found 7059 people in the images
False Positives (3945): The model mistakenly identified 3945 backgrounds as people
False Negatives (2519): The model mislabeled 2519 people as background.
True Negatives: there were no background images during training. As all the images were of persons.

3.1.4. Visual result

The visual output from the model, as shown in the screenshot of a video frame, demonstrates the model's precision in real-time pose estimation. Key points are plotted on individuals in a public transport setting, with lines connecting various points to detect body posture.

Figure 5: *poses of persons sitting in a bus*

3.2. Results for FasterRCNN (Chairs Detection)

After training, we evaluated the model's performance using several metrics, including visualization of predictions, generation of a classification report, and construction of a confusion matrix. Additionally, we visualized the training losses to understand the model's learning process better.

We loaded the best model checkpoint and ran inference on test images to visualize the predictions.

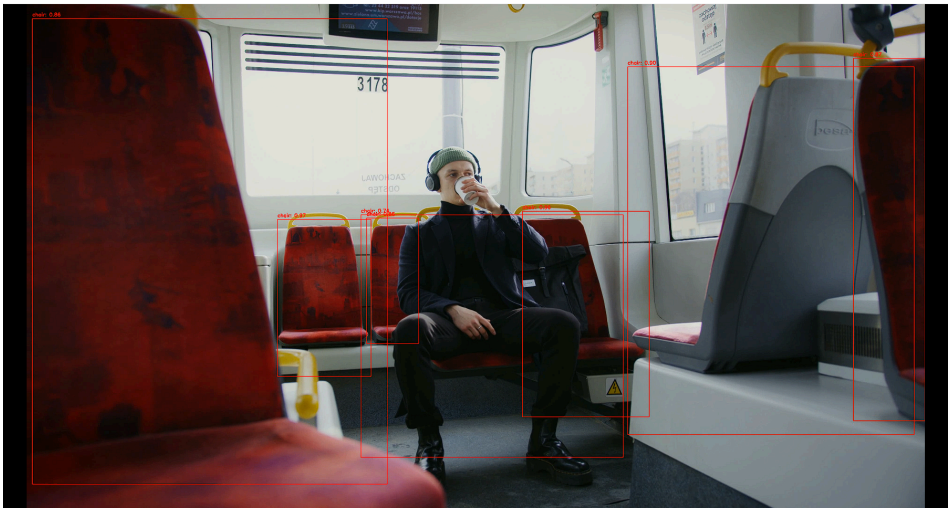

Figure 6: *Chairs Detection Using FasterRCNN*

3.2.1. Faster R-CNN Metrics Summary

The classification accuracy improves significantly from the early stages of training, starting at 0.595 at iteration 9 and reaching up to 0.943 by iteration 5000. The rate of false negatives decreases over time, indicating that the model is missing fewer instances as training progresses. This could be due to the model learning more features that are indicative of the objects it needs to detect. The total

loss decreases as the model trains, which is expected and indicates overall improvement in the model's predictions across all parameters. This is a good sign that the training process is effective and that the model is converging towards a state of lower error rates.

Iteration	Cls Accuracy	False -ve	Loss (Box Reg)	Loss (Cls)	Loss (RPN Cls)	Loss (RPN Loc)	Total Loss
9	0.595	0.308	0.6601	0.6698	0.0530	0.0107	1.419
999	0.908	0.302	0.383	0.205	0.028	0.011	0.611
1999	0.945	0.158	0.301	0.163	0.025	0.010	0.563
2999	0.943	0.173	0.293	0.134	0.013	0.006	0.438
3999	0.935	0.171	0.291	0.144	0.005	0.005	0.500
5000	**0.943**	**0.144**	**0.318**	**0.145**	**0.006**	**0.008**	**0.499**

3.2.2. Loss Function

- **Box Regression Loss**: Measures the accuracy of the predicted bounding boxes against the ground truth boxes. It ensures that the predicted locations of objects match their actual positions, improving the precision of object localization. At iteration **5000**, the box regression loss is **0.318**, indicating the model's accuracy in predicting the bounding box locations.
- **Classification Loss**: Measures the accuracy of the predicted class labels against the ground truth labels. It ensures that the model correctly identifies the objects in the image. At iteration **5000**, the classification loss is **0.145**, reflecting the model's performance in classifying detected objects.
- **RPN Classification Loss**: Measures the accuracy of the Region Proposal Network (RPN) in classifying anchors as either foreground (object) or background (non-object). It Helps the model generate high-quality region proposals, which are essential for detecting objects accurately in the subsequent stages. At iteration **5000**, the RPN classification loss is **0.006**, indicating the RPN's ability to classify anchors correctly.
- **RPN Localization Loss**: Measures the accuracy of the predicted anchor box locations against the ground truth boxes. It ensures that the region proposals are precisely localized, providing a good starting point for the final object detection stage. At iteration **5000**, the RPN localization loss is **0.008**, indicating the precision of the anchor box locations.

3.2.3. Evaluation Results:

To better understand the training process, we visualized the loss curves over iterations, which provided insights into the model's learning dynamics.

The graph shows a decline in all forms of loss at the beginning of the training process, particularly noticeable in the total loss. This initial drop is common as the model quickly learns, correcting major mistakes in its predictions. After the initial improvements, the losses begin to stabilize, starting to converge towards an optimal set of weights where making further improvements becomes difficult.

Both the general classification loss and the RPN classification loss remain low throughout the training, which indicates that once the model learns to propose regions accurately, it maintains this ability consistently.

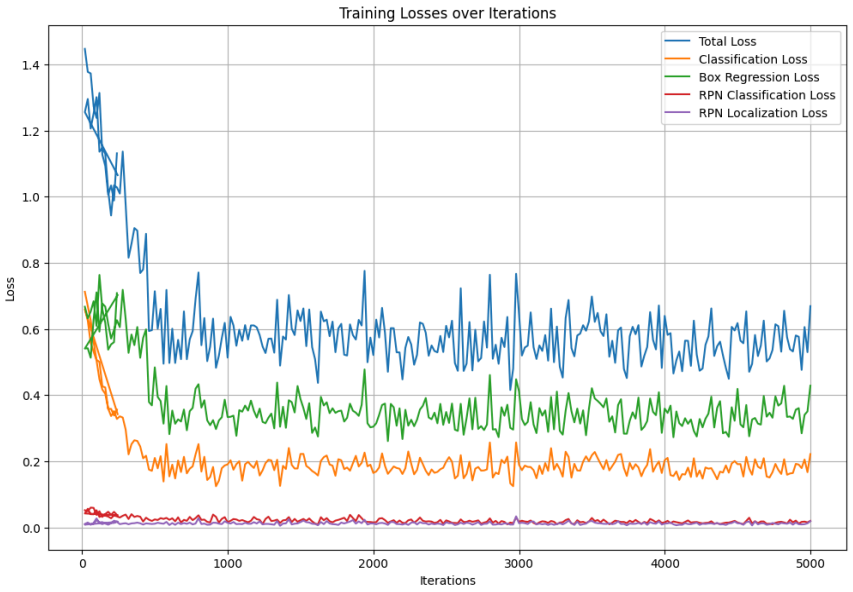

Figure 7: *Training losses for FasterRCNN*

Metric	**AP**	**AP50**	**AP75**	**APs**	**APm**	**APl**
Results	0.531	2.395	0.020	0.312	0.644	0.872

3.3. Results for Yolov8 Object Detection

The Yolov8 model identifies and outlines chairs within a public transportation environment. The image shows the detection of multiple chairs, each one highlighted with a green box.

Figure 8: *Chairs Detection Using Yolov8*

3.3.1. Yolov8 Metrics Summary

This table represent the performance metrics of the model over 30 epochs of training, showing how it learns and adapts.

Both training losses (box and cls) show a general decline as the epochs progress, indicating that the model is learning during training and its improving its predictions.

There is an upward trend in both precision and recall values, so the model identifies an higher percentage of true positive results (precision) and also successfully retrieves a bigger fraction of the actual positive samples (recall).

The increase in both mAP50 and mAP50-95 scores across the epochs suggests that the model is improving its ability to match the ground truth bounding boxes with higher precision at various thresholds.

The validation losses for both box and classification initially show fluctuations but gradually decrease, reflecting that the model is generalizing well to new data and not just fitting to the training dataset (reduced overfitting).

Epoch	Train Box Loss	Train Cls Loss	Precision (B)	Recall (B)	mAP50 (B)	mAP50-95 (B)	Val Box Loss	Val Cls Loss
1	1.5182	2.1988	0.31247	0.28029	0.22946	0.12336	1.7968	2.3818
5	1.5949	2.0499	0.41209	0.32105	0.311	0.17812	1.5717	1.9576
10	1.4628	1.8174	0.50964	0.35008	0.3849	0.23106	1.4446	1.8049
15	1.3836	1.6829	0.56237	0.40201	0.43675	0.26894	1.4023	1.6038
20	1.3336	1.5726	0.59189	0.41374	0.46695	0.29633	1.3363	1.5102

25	1.27	1.411	0.5765	0.43216	0.48211	0.30815	1.3312	1.4649
30	**1.2184**	**1.3062**	**0.55788**	**0.47557**	**0.49133**	**0.31464**	**1.3295**	**1.4622**

3.3.2. Loss Function

- **Box Regression Loss**:
  It measures the accuracy of the predicted bounding boxes against the ground truth boxes. It Ensures that the model precisely locates objects within the image. Accurate bounding boxes are crucial for determining the exact position of detected objects. Ranging from **1.5182** at epoch 1 to **1.2184** at epoch 30, indicating improvement in bounding box predictions over epochs.
- **Classification Loss**:
  Measures the accuracy of the predicted class labels against the ground truth labels. Ensures that the model correctly identifies the type of object detected within each bounding box. Accurate classification is essential for distinguishing between different object classes. Starting at **2.1988** at epoch 1 and decreasing to **1.3062** by epoch 30, showing enhanced classification accuracy over time.
- **Distribution Focal Loss (DFL)**:
  Focuses on hard-to-classify examples by assigning higher loss to misclassified examples and lower loss to well-classified ones. Helps improve the accuracy of the model by making it more sensitive to difficult examples, thereby enhancing the model's overall robustness and performance. This is integrated into the overall loss, improving the model's handling of challenging examples.

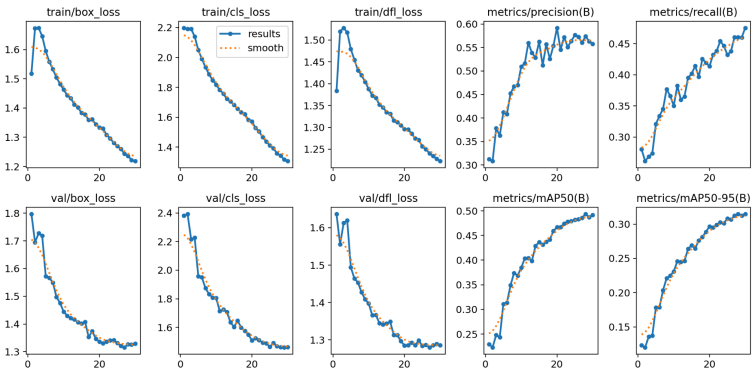

Figure 9: *Training and Validation Metrics for Pose Estimation*

3.3.3. Confusion matrix

This confusion matrix distinguishes between “chair” and “background” classes. The table suggests that the model can identify the chairs, but has a tendency to miss a significant number of chairs (high FN) and mistakenly label other objects as chairs (high FP).

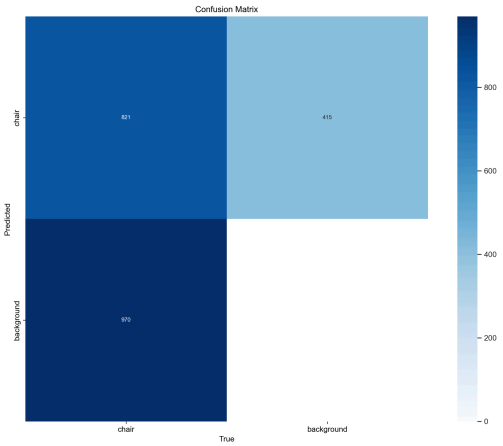

Figure 10: Confusion Matrix Of Chairs Detection using Yolov8

True Positives (821): The model correctly found 821 chairs in the images
False Positives (415): The model mistakenly identified 415 backgrounds as chairs
False Negatives (970): The model mislabeled 970 chairs as background.
True Negatives: there were no background images during training. As all the images were of chairs.

3.4. Faster R-CNN vs. YOLO

Below is the comparison of faster-RCNN and YOLO. Showing all the training metrics.

3.4.1. Faster R-CNN Metrics Summary

- **Cls Accuracy:** Improved from 0.595 at iteration 9 to **0.943** at iteration 5000
- **False Negatives:** Decreased from 0.308 at iteration 9 to **0.144** at iteration 5000
- **Losses:**
  - **Box Regression Loss:** decreased from 0.6601 to **0.318**
  - **Classification Loss:** reduced from 0.6698 to **0.145**
  - **RPN Classification Loss:** Decreased from 0.0530 to **0.006**
  - **RPN Localization Loss:** starting at 0.0107 and ending at **0.008**
  - **Total Loss:** Significantly decreased from 1.419 to **0.499**
- **Precision and Recall:**
  - **AP:** 0.531
  - **AP50:** 2.395
  - **AP75:** 0.020

- **APs:** 0.312
- **APm:** 0.644
- **APl:** 0.872

3.4.2. *YOLOv8 Metrics Summary*

- **Losses:**
  - **Train Box Loss:** Decreased from 1.5182 to **1.2184**
  - **Train Classification Loss:** Decreased from 2.1988 to **1.3062**
  - **Val Box Loss:** Decreased from 1.7968 to **1.3295**
  - **Val Classification Loss:** Decreased from 2.3818 to **1.4622**
- **Performance Metrics:**
  - **Precision (B):** Increased from 0.31247 to **0.55788**
  - **Recall (B):** Increased from 0.28029 to **0.47557**
  - **mAP50 (B):** Increased from 0.22946 to **0.49133**
  - **mAP50-95 (B):** Increased from 0.12336 to **0.31464**

Comparing both Faster R-CNN and YOLOv8, Faster R-CNN shows more improvements in key metrics such as classification accuracy, total loss, and AP scores. The continuous decrease in various loss metrics shows its learning capabilities. Moreover, the high accuracy and lower false negatives indicate better precision in detecting and classifying objects.

3.5 SMART SEAT

In the smart seat project we have combined the FasterRCNN model for chairs detection with Yolov8 pose estimation model. First FasterRCNN will detect all the chairs and draw green boxes around them. The pose estimation model detects the key points of a person and with the help of an algorithm it estimates the pose whether the person is in sitting position or standing.

Afterwards, the algorithm checks if the person keypoints are overlapping on the chair bounding box if yes then it further checks another condition about the pose that if it's marked as sitting, the chair is considered as occupied or else it's unoccupied.

The final output includes a summary indicating the number of detected chairs, how many are occupied and the number of detected persons. The results are displayed on the frame and saved to an output video file. Below is a screenshot of a video, showing that the chair is occupied in red. Other chairs are marked as unoccupied.

Figure 11: *Smart Seat (Chair occupied by looking at the pose and the chair bounding box)*

4. CONCLUSION

In our Smart Seat project, we successfully integrated advanced AI models to detect and analyze seat occupancy in real-time. By combining Faster R-CNN for accurate chair detection with YOLOv8 for pose estimation, we created an effective system that can distinguish between occupied and unoccupied seats. Our methodology significantly improves upon traditional seat occupancy detection methods, providing a robust solution for safety and efficiency in various environments.

5. BIBLIOGRAPHY

1. Gangwani, D., & Gangwani, P. . "Applications of Machine Learning and Artificial Intelligence in Intelligent Transportation System: A Review."
2. COCO Consortium, "COCO Dataset," https://cocodataset.org/#download
3. Amikelive, "What Object Categories Labels Are in COCO Dataset?" https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/
4. Ultralytics, "COCO Dataset YAML for Object Detection," https://docs.ultralytics.com/datasets/detect/coco/#dataset-yaml
5. Ultralytics, "YOLOv8 Object Detection," https://docs.ultralytics.com/modes/detect/
6. Ultralytics, "YOLOv8 Pose Estimation," https://docs.ultralytics.com/tasks/pose/
7. PyTorch, "Faster R-CNN Model," https://pytorch.org/vision/main/models/faster_rcnn.html
8. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *University of Washington*, *Allen Institute for AI*, *Facebook AI Research*.
9. I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," 2012 Proceedings of the 35th International Convention MIPRO, Opatija, Croatia, 2012, pp. 1725-1730.
10. CUDA and cuDNN: https://developer.nvidia.com/cudnn