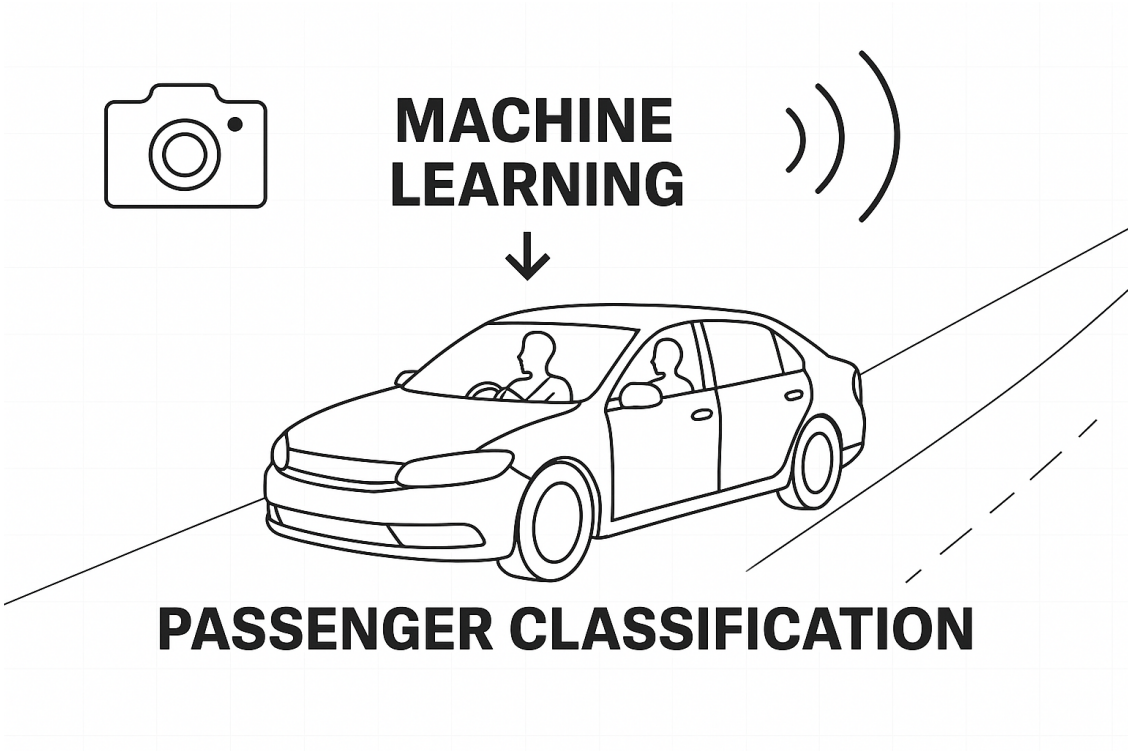
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Machine Learning for In-Vehicle Occupant Classification Using Radar Heatmaps and 3D Camera-Based Keypoints

Master's thesis in Computer Science and Engineering

Abdullah Daboul

Mohammad Kassem

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Machine Learning for In-Vehicle Occupant Classification Using Radar Heatmaps and 3D Camera-Based Keypoints

Abdullah Daboul

Mohammad Kassem

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Machine Learning for In-Vehicle Occupant Classification Using Radar Heatmaps and 3D Camera-Based Keypoints
Mohammad Kassem, Abdullah Daboul
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Supervisor: Yinan Yu, Department of Computer Science and Engineering
Advisor: Jonas Ekdahl, Volvo Cars
Examiner: Jean-Philippe Bernardy, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Abstract

This thesis investigates how machine learning (ML) models can classify vehicle occupants as adults, forward-facing children, or rear-facing children based on interior sensor data. The goal is to improve safety systems, such as airbag deployment, by identifying the occupant type using radar and camera keypoint data. We evaluate radar-based heatmaps using a deep learning model and 3D keypoints derived from a camera using ML models, comparing their performance and data requirements.

Our primary method combines a Convolutional Neural Network (CNN) with a Long Short-Term Memory (LSTM) network to analyze sequences of radar heatmaps. The architecture is trained and validated using data from 61 passengers captured in a stationary vehicle with synchronized radar and camera sensors. Additional experiments use classical ML models on body keypoints extracted from the camera. In addition. we apply incremental training and extrapolate learning curves.

The best radar-only model, based on ResNet-50 + LSTM, achieved a macro F1-score of 0.867 and classification accuracy of 89.2%, with strong per-class performance (F1 = 0.831 for rear-facing children, 0.934 for forward-facing children, and 0.836 for adults). A smaller TinyCNN + LSTM model maintained competitive results while reducing memory demands. Keypoint-only classifiers reliably separated adults from forward-facing children with F1-scores above 0.96, though rear-facing cases were excluded.

Overall, the results show that deep learning models can accurately classify occupant types using either radar or camera alone. We also show that more data will significantly improve performance, especially for underrepresented classes such as rear-facing children. By extrapolating learning curves, we provide practical estimates of data requirements, enabling informed decisions about dataset expansion.

Keywords: neural networks, radar, machine learning, clustering, heatmaps, keypoints, camera, CNN–LSTM.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, **Yinan Yu**, at the Department of Computer Science and Engineering, for the invaluable guidance, feedback, and support throughout this thesis. Her insights and encouragement played a crucial role in shaping the direction and quality of this work.

We are also deeply thankful to our industrial advisor, **Jonas Ekdahl** from **Volvo Cars**, for his expertise, technical input, and practical perspective, which grounded the research in real-world challenges and motivated many of the improvements explored in this study.

We would like to thank **Jean-Philippe Bernardy**, our examiner, for his thoughtful comments and evaluation, which helped refine the final report.

Lastly, We are grateful to the broader teams at Chalmers and Volvo Cars who provided tools, data, and an inspiring research environment. This project would not have been possible without their support.

Mohammad Kassem & Abdullah Daboul, Gothenburg, 2025-06-15

Contents

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Purpose . . . 1
1.2 Limitations . . . 2
1.3 Ethical Considerations . . . 2
1.4 Outline of the Thesis . . . 3

2 Theory **5**
2.1 Radar . . . 5
2.1.1 Generation of the Range-Azimuth Map . . . 5
2.2 Occupant classification systems . . . 7
2.2.1 Weight Sensing . . . 7
2.2.2 Camera-based sensors . . . 7
2.2.3 Radar-based sensors . . . 7
2.3 Machine learning . . . 8
2.3.1 Supervised Learning Methods . . . 9
2.3.2 Unsupervised Learning Methods . . . 10
2.4 Deep Neural Networks . . . 11
2.4.1 Convolutional Neural Networks (CNNs) . . . 11
2.4.2 Long Short-Term Memory (LSTM) Networks . . . 12
2.4.3 Hybrid CNN–LSTM Architectures . . . 13
2.5 Transfer Learning . . . 14
2.5.1 ResNet Architecture . . . 14
2.5.2 Transfer Learning Approaches . . . 15

3 Methods **17**
3.1 Dataset . . . 17
3.1.1 Building the Initial Labeled Dataset from ROSBags . . . 18
3.1.2 Preprocessing of Azimuth Radar Heatmaps . . . 19
3.1.3 Preprocessing of 3D Keypoints . . . 19
3.2 Architecture Design . . . 22
3.2.1 CNN–LSTM . . . 22
3.3 Evaluation Methods . . . 22

3.3.1 Confusion Matrix . . . 23
3.3.2 Accuracy . . . 23
3.3.3 Precision . . . 23
3.3.4 Recall . . . 24
3.3.5 F1-Score . . . 24
3.3.6 Macro-Averaging for Multi-Class Classification . . . 24
3.4 Cross-Validation Setup and Data Scaling . . . 25
3.4.1 CNN–LSTM on Radar Heatmaps . . . 25
3.4.2 ResNet + LSTM (All 3D Keypoints) . . . 25
3.4.3 ResNet + LSTM (3D Body Metrics) . . . 25
3.4.4 Supervised Models on Heatmaps / Keypoints . . . 25
3.5 Unsupervised Clustering of 3D Keypoint Features . . . 26
3.6 Estimating Training Data Requirements via Performance Extrapolation 27
3.6.1 Regression-Based Estimation Framework . . . 27
3.6.2 Performance Targeting and Error Analysis . . . 27
3.6.3 Motivation from Prior Work . . . 28
3.7 Cluster-Based Risk Assessment Framework . . . 28
3.7.1 Sequence-Level Risk Analysis Workflow . . . 28
3.7.2 Risk Label Assignment . . . 29

4 Results **31**
4.1 Evaluation of the Deployed Volvo Cars Model . . . 31
4.2 Unified Training Framework . . . 32
4.2.1 Data Splitting and Training Ratios . . . 33
4.2.2 Radar-Only vs. Multimodal (Radar + Keypoints) Models . . . 33
4.2.3 Traditional Supervised Classifiers . . . 34
4.2.4 Cluster-Based Risk Assessment . . . 34
4.3 Radar Heatmap Classification with CNN–LSTM Architectures . . . 34
4.3.1 Comparative Model Performance . . . 34
4.4 Multimodal CNN–LSTM Experiments . . . 36
4.4.1 Comparative Model Performance . . . 36
4.5 Supervised Models on Heatmaps . . . 37
4.5.1 Comparative Model Performance . . . 37
4.6 3D Keypoint-Based Subsequence Classification Child-Forward and Adult . . . 38
4.6.1 Comparative Model Performance . . . 38
4.6.2 Feature Importance Analysis . . . 39
4.7 Clustering Analysis of 3D Keypoints . . . 39
4.7.1 Body Metrics: Torso, Shoulder, and Hip . . . 40
4.7.2 Clustering on All 3D Keypoints . . . 40
4.7.3 Error Analysis of Misclassified Sequences . . . 41
4.8 Regression-Based Sample Size Estimation . . . 43
4.8.1 Estimated vs. Actual Additional Frames . . . 43
4.9 Impact of Cluster-Based Risk Assessment on Model Performance . . . 45
4.9.1 Training Strategy Overview . . . 45
4.9.2 Cross-Fold Performance Comparison . . . 46

4.9.3 Misclassification Pattern Analysis 46

5 Discussion **49**
5.1 Motivation from Deployed Model Evaluation 49
5.2 Radar Heatmap Classification with CNN–LSTM 50
5.3 Multimodal CNN–LSTM (Heatmaps + Keypoints) 50
5.4 Supervised Learning on Flattened Radar Heatmaps 51
5.5 3D Keypoint-Based Subsequence Classification 52
5.6 Clustering Analysis of 3D Keypoints 52
5.7 Regression-Based Sample Size Estimation 52
5.8 Risk-Aware Oversampling via Clustering 53

6 Conclusion **55**

Bibliography **57**

A Appendix 1 **I**

List of Figures

2.1 Overview of the radar signal processing pipeline, starting from the front-end hardware and culminating in a spatial visualization of the scene [3]. 6

2.2 Radar mounted to the ceiling of the cabin for in-vehicle occupant classification, from [8]. 8

2.3 Connectivity of the pooling and kernels in CNNs [19] 12

2.4 Internal connectivity of an LSTM memory cell, illustrating the forget, input, and output gates that regulate the flow of information across time steps [19]. 12

2.5 Schematic of a hybrid CNN–LSTM architecture: a frame-wise CNN extracts spatial features at each time step, and the resulting feature sequence is processed by an LSTM to model temporal dependencies [21]. 14

3.1 **Mean 3D Keypoint Quality per Class.** Each cell shows the average confidence score (ranging from 0 to 1) for a specific keypoint, aggregated by true class label. Class 4 (Child-Rearward) exhibits consistently low quality across nearly all keypoints, motivating its exclusion from unsupervised clustering. 21

3.2 CNN–LSTM network architecture 22

4.1 Confusion Matrix for the Occupant Classification Deployed Model. . . 32

4.2 Top-15 Most Influential 3D Keypoint Features for Classification Models. Each plot shows the normalized importance scores across models that support feature attribution. 39

4.3 Clustering using body metrics: torso length, shoulder width, and hip width. Clusters correspond to Adult and Child-Forward classes. . . . 40

4.4 UMAP projection of normalized 3D keypoints. Top-left: true class labels; top-right: KMeans assignments; bottom: misclassified points (black). 41

4.5 Mean 2D Skeleton Comparison (X-Y View). This plot compares average 2D skeleton projections for three groups: misclassified adults (red), correctly classified children (blue), and correctly classified adults (green). Each skeleton is plotted using key anatomical landmarks, where the lower points represent hip joints, the extended lateral points form arms (elbows and wrists), and the upper structure includes shoulders, nose, eyes, and ears. The visual separation in scale and proportions particularly in limb length and shoulder spread helps illustrate the spatial features the model uses and struggles with when classifying body types. 42
4.6 Comparison of body geometry metrics across groups. Misclassified adults lie between true children and true adults in torso and shoulder dimensions. 42
4.7 Regression-based extrapolation of F1-score as a function of training data ratio for Child-Rearward, Child-Forward, and Adult. The target F1-score in figure is 0.945 43
4.8 Macro F1-score across five folds. Risk-based oversampling (*with_review_emphasis*) outperforms both the baseline and random oversampling. Removing the error ratio criterion reduces performance. 46
4.9 Misclassification breakdown across folds. Most errors occur where the predicted label matches a similar (but incorrect) training sample. . . 47

List of Tables

2.1 Comparison of ResNet architectures on ImageNet dataset 15

4.1 Classification Report for the Occupant-Classification Model 32
4.2 Average Per-Class F1-scores Across Training Sizes and Models (Mean ± Std) 35
4.3 Macro F1-socre and Accuracy Across Training Sizes for Each Model (Mean ± Std) 36
4.4 Average Per-Class F1-scores Across Training Sizes for Multimodal Models (Mean ± Std) 36
4.5 Macro F1-score and Accuracy Across Training Sizes for Multimodal Models (Mean ± Std). 37
4.6 Cross-Validation Summary for Supervised Models (Heatmap Subsequences, Outliers Included). Scores represent Macro F1-score (mean ± std) across five folds. 37
4.7 Average Per-Class F1-scores Across Training Sizes for Supervised Models (Heatmap Subsequences) 38
4.8 Models Performance: Macro and Per-Class F1-Scores 38
4.9 Estimated vs. Actual Required Ratios – Child-Rearward 44
4.10 Estimated vs. Actual Required Ratios – Child-Forward 45
4.11 Estimated vs. Actual Required Ratios – Adult 45

1

Introduction

Modern vehicle safety systems like airbags and seatbelt pretensioners are designed to protect occupants during collisions. However, these systems are optimized for adult bodies and can pose serious risks to small children. In particular, front-seat airbags can cause fatal injuries to children seated in rear-facing car seats. Between 1989 and 1997, 44 children in the U.S. were killed by airbag deployments that would not have been fatal otherwise [1].

To mitigate these risks, occupant classification systems (OCS) aim to detect and categorize passengers by distinguishing adults from children. Based on this classification, vehicle safety features such as airbag deployment and seatbelt pretensioning can be adapted to better match the occupant's size and seating orientation. This thesis explores how ML models can classify front-seat occupants into three categories: rear-facing child, forward-facing child, and adult. The goal is to improve safety outcomes using minimal labeled training data.

Two types of in-cabin sensor data are evaluated: radar-based heatmaps and camera-based 3D body keypoints. The radar modality captures sequences of spatial heatmaps generated from azimuth reflections, which are processed using a Convolutional Neural Network (CNN) followed by a Long Short-Term Memory (LSTM) network. The camera modality uses 3D pose estimation to extract keypoints, which are then analyzed with classical ML models. Due to limited visibility, the rear-facing class is excluded from the camera-based pipeline.

The study also examines how model performance scales with training data size and introduces a clustering-based oversampling method to improve accuracy for minority classes. Finally, learning curves are extrapolated to estimate how much data is required to reach target performance levels.

1.1 Purpose

The primary aim of this thesis is to comprehensively evaluate and improve in-vehicle occupant classification using radar-based heatmaps and multimodal sensor data. The work explores and compares a range of approaches including deep CNN–LSTM architectures, classical ML models, and unsupervised clustering methods to classify forward-facing children, rearward-facing children, and adults. In addition to model development, the thesis investigates strategies for minimizing data requirements

through sample size extrapolation and enhances training effectiveness via a novel cluster-based risk-aware oversampling technique. The study also critically analyzes the limitations of an existing production model deployed in a vehicle, motivating the design of custom pipelines for error diagnosis, feature analysis, and performance improvement. Ultimately, this work aims to advance occupant classification capabilities while balancing model accuracy, data efficiency, and deployment feasibility.

To achieve these aims, this thesis seeks to answer the following research questions:

1. How accurately can deep learning models classify vehicle occupants as rear-facing children, forward-facing children, or adults using only radar heatmap sequences alone, and how does performance change when combining radar heatmaps with keypoint features as multimodal inputs? Additionally, how does classification accuracy scale with varying training dataset sizes?
2. What is the occupant classification performance of classical ML models trained on 3D body keypoints extracted from camera data?
3. Approximately how much labeled training data is required to achieve target F1-scores for each occupant class (Adult, Child-Forward, Child-Rearward), and how reliably can these data volume requirements be estimated using regression-based extrapolation of learning curves?

1.2 Limitations

A key limitation of this work is the dependency on a proprietary dataset provided by an automotive company, which restricts the breadth of data variability and poses challenges for reproducibility. Moreover, the dataset is imbalanced. There are significantly more samples for certain occupant classes than others, which may affect the model's performance. Additionally, the current study focuses exclusively on occupant classification in a stationary vehicle environment, so the findings may not directly extend to dynamic driving conditions. Finally, while the integration of advanced deep learning methods is explored, the final models are intended as proof-of-concept rather than production-ready solutions.

1.3 Ethical Considerations

All data used in this thesis were collected and provided by Volvo Cars. The original dataset includes personally identifiable information, such as images and biometric attributes such as height and weight. However, the analysis presented in this thesis was conducted exclusively on abstracted data representations, specifically radar heatmaps and 3D keypoints, ensuring that no personally identifiable information was used in the research. Participants involved in the data collection were informed about the purpose of the study and the intended use of the data for research and development. No new data was collected by the author, and all processing was performed within the bounds of responsible data handling and in compliance with the data usage agreement established with Volvo Cars.

1.4 Outline of the Thesis

The remainder of this thesis is organized as follows: Chapter 2 presents the theoretical foundations relevant to this work, including radar sensing, ML, deep neural networks, and transfer learning. Chapter 3 describes the dataset construction and preprocessing, the architecture design, and the evaluation methods, including both supervised and unsupervised approaches. Chapter 4 presents the experimental results across different models and modalities, as well as performance evaluations, clustering analyses, and sample size estimation. Finally, Chapter 5 discusses the main findings in the context of existing research, highlights limitations, and concludes the thesis with suggestions for future work.

2

Theory

This chapter introduces the theoretical foundations relevant to the thesis. It begins with an overview of radar systems, including the process for generating range-azimuth maps. This is followed by a discussion on ML, covering both supervised and unsupervised learning techniques. The chapter then explores deep neural networks, focusing on Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and hybrid CNN–LSTM architectures. Finally, the concept of transfer learning is presented, with an emphasis on the ResNet architecture and common transfer learning approaches.

2.1 Radar

Radar (Radio Detection and Ranging) is a technological system that uses electromagnetic waves to detect and locate objects within its field of view [2]. It operates by transmitting electromagnetic signals, such as modulated sine waves or pulses, and processing the reflected signals, known as echoes, to determine the characteristics of objects in its surroundings. Radar is particularly effective in conditions where visual observation is limited, such as darkness, fog, or heavy precipitation, making it indispensable in applications ranging from air traffic control to autonomous vehicle safety.

A radar system consists of a transmitter, a receiver, and an antenna. The transmitter generates electromagnetic waves, which the antenna radiates into the environment. When these waves encounter an object, a portion of the energy is scattered, and some of it is reflected back toward the radar system. The receiving antenna captures this backscattered signal, which is then processed to extract information about the object's location, size, and movement. By analyzing factors such as time delay, frequency shift (Doppler effect), and signal strength, radar can accurately determine an object's distance, velocity, and other characteristics.

2.1.1 Generation of the Range-Azimuth Map

After the radar has captured the echo data, the next critical step in radar signal processing is to generate a spatial representation of the scene. One common output is the range-azimuth map, which provides a polar view of the radar's field of view by mapping the signal returns according to their range and azimuth angle [3].

The overall processing pipeline, from the analogue front-end to the visualization output, is illustrated in Figure 2.1. The figure shows how raw signals are successively digitized, processed, and ultimately converted into a spatially meaningful format such as a Range-Doppler-Azimuth map.

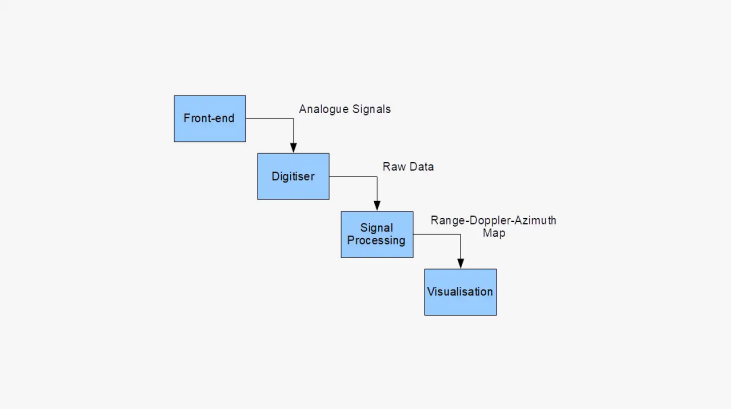


Figure 2.1: Overview of the radar signal processing pipeline, starting from the front-end hardware and culminating in a spatial visualization of the scene [3].

In parallel, the radar's field of view is divided into several angular sectors (or "spokes"). Each sector represents a specific azimuth angle relative to the radar. By processing the signal returns for each angular sector independently, the radar system can determine the spatial distribution of targets within its coverage area.

To create the range-azimuth map, the following conceptual steps are performed:

1. **Segmentation:** The continuous raw data is segmented into frames, each corresponding to one complete pulse or ramp interval. Within each frame, the samples are grouped according to the specific angular sector from which they originate.

2. **Range Processing:** For each group of samples, a transformation using a Fast Fourier Transform (FFT) is applied to convert the time-domain signal into a frequency-domain representation. This transformation isolates the beat frequency that directly correlates with the time delay and therefore the range to the target.

3. **Angular Assignment:** Simultaneously, the radar associates each processed segment with a corresponding azimuth angle based on the known scan pattern or beam-steering characteristics. This mapping is derived from the spatial configuration of the antenna(s) and the radar's scanning strategy.

4. **Visualization:** The processed data is then assembled into a two-dimensional map. In this range-azimuth map, one axis represents the distance (range) to the target while the other axis represents the azimuth angle. The magnitude of the radar return (often represented in logarithmic units) is used to indicate the strength of the reflection for each cell in the map.

2.2 Occupant classification systems

Occupant classification systems (OCS) have evolved significantly over the past decades, transitioning from simple mechanical sensors to sophisticated multimodal systems integrating advanced sensing technologies and ML algorithms. This progression aims to enhance vehicle safety by accurately detecting and classifying occupants, particularly distinguishing between adults and children, to inform airbag deployment decisions.

2.2.1 Weight Sensing

Early OCS employed weight-based sensing technologies, such as strain gauge load cells integrated into the seat structure [4], to estimate occupant weight and center of gravity. These systems classified occupants into discrete weight categories and disabled or modified airbag deployment when the estimated weight fell below a predefined threshold, typically around 25–30 kg, to reduce the risk of injury to small occupants, including children. According to the U.S. National Highway Traffic Safety Administration (NHTSA), a human child weighing between 21 and 25.6 kg and measuring 114 to 124.5 cm in height may be used in place of a 6-year-old crash test dummy in compliance testing under sections S23.2 and S23.5.1 of FMVSS No. 208 [5].

2.2.2 Camera-based sensors

Early OCS based on cameras often used simple ML techniques. Malneedi and Soman [6] developed a lightweight approach using infrared imaging combined with Haar cascade face detection and a Support Vector Machine (SVM) to classify occupants as adults or children. Their system achieved over 99% accuracy on controlled tests and demonstrated that classical methods can offer reliable performance in constrained conditions. However, such models often struggle when transferred across different vehicles due to variations in cabin design and background features. To overcome this, Dias Da Cruz et al. [7] proposed an autoencoder-based method that learns to remove background information and focus solely on occupant features. Their model reconstructs clean cabin images during training and classifies occupants from a more stable internal representation. Tested on synthetic and real infrared images, this approach generalized better across different vehicle interiors without needing retraining, addressing a key challenge in real-world deployment.

2.2.3 Radar-based sensors

In [8], Van Marter et al. proposed a deep learning approach for in-vehicle multi-occupant detection and classification using a 60-GHz mmWave FMCW radar. Contrary to the more common use of radar for exterior sensing, their setup employed an in-cabin radar mounted to the ceiling of the vehicle interior, directly facing the passenger seats. This configuration enabled the radar to scan the interior space and collect reflections from seated occupants. Raw radar signals were processed

through multipass Constant False Alarm Rate (CFAR) thresholding and zoom-in Capon beamforming to generate sparse 3-D point clouds, capturing range, azimuth, elevation, and velocity while reducing computational complexity and data storage compared to full 3-D heatmaps. The dataset comprised 657 trials conducted in a five-seater vehicle with 27 participants (15 adults, 11 children, and one realistic baby doll), under realistic in-cabin conditions. Their network design incorporated attention mechanisms to handle the irregular structure of point cloud data and was trained to perform localized detection, classify each seat (as adult, child, or baby), and detect child presence. The system achieved 95.6% accuracy in seat-wise detection, 88.7% in occupant classification, and 88.4% for smart CPD. Fig. 2.2 from their work illustrates the radar placement inside the vehicle.

Figure 2.2: Radar mounted to the ceiling of the cabin for in-vehicle occupant classification, from [8].

In [9], a 77-GHz mm-wave MIMO FMCW radar system was employed to capture RF reflections within the vehicle. Raw radar data were processed using a Capon beamformer combined with delay-and-sum imaging to o create 3D heatmaps that represent the range, azimuth, and elevation of reflected signals. The dataset comprised 780 scenarios recorded with a commercially available MIMO radar, where each transmission was captured by 20 receiving antennas. Data were collected in stationary vehicles occupied by five passengers, including infants and children. For classification, the study proposed a deep learning model based on a 2-D CNN–LSTM architecture. The CNN component extracts spatial features from the processed 3-D heatmaps, while the LSTM captures temporal dynamics across successive frames. For the detection of unattended infants/children in vehicles, an accuracy of over 95% was achieved. For counting passengers and identifying their occupied seats, the accuracy was 89%, and for classifying passengers (adult vs. child), the accuracy was over 74%.

2.3 Machine learning

Machine Learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing computational models capable of improving their performance through experience rather than relying on explicitly programmed instructions [10]. At its core, ML involves designing algorithms that can identify patterns in data and use those patterns to make predictions, decisions, or classifications. The fundamental premise of ML is that systems can learn from historical data, refine their models based on new inputs, and generalize their knowledge to unseen scenarios.

A ML model consists of an algorithm equipped with a set of parameters that adapt during the learning process to optimize performance for a given task. The learning process is typically divided into two stages: training and evaluation [10]. During training, the model learns from a dataset known as the training set, where the relationships between inputs and outputs (if applicable) are established. The model's ability to generalize is assessed using a separate test set, which consists of previously unseen data points. Various performance metrics, such as accuracy, precision, recall, and loss functions, are used to evaluate how well the model has learned and to identify areas where adjustments are needed.

ML is broadly categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning [10]. Supervised learning involves learning from labeled data, where the algorithm attempts to map inputs to known outputs. Unsupervised learning, on the other hand, works with unlabeled data and seeks to identify hidden patterns or structures within the dataset. Reinforcement learning differs from both in that an agent interacts with an environment and learns by receiving feedback in the form of rewards or penalties.

Within supervised and unsupervised learning, various techniques and models exist to address specific tasks. Supervised learning includes methods such as logistic regression, support vector machines (SVMs), decision trees, and ensemble models like XGBoost [10]. In contrast, unsupervised learning encompasses clustering techniques such as k-means and hierarchical clustering. In recent years, deep learning, a subset of ML based on artificial neural networks, has gained significant attention due to its ability to handle complex tasks such as image recognition, natural language processing, and autonomous systems.

The subsequent sections will explore these categories in greater detail, beginning with supervised learning and its core methodologies.

2.3.1 Supervised Learning Methods

XGBoost

XGBoost is a ML method based on gradient boosting, where models are built sequentially by training new decision trees to correct the residual errors of previous ones. Decision trees partition the feature space through splits, grouping data into subsets that make predictions. XGBoost minimizes a regularized objective that balances prediction error and model complexity to reduce overfitting, and uses second-order derivatives, approximate split finding, missing value handling, and parallel computation for efficient training [11].

K-Nearest Neighbors K-Nearest Neighbors (kNN) is a simple, non-parametric ML method used for classification and regression tasks. It makes predictions based on the labels or values of the closest training examples in the feature space. Given a new input, kNN identifies the k training instances with the smallest distance to the input, typically using Euclidean distance. For classification, the output is usually the majority label among the neighbors, while for regression, it is the average of their target values. The method does not involve explicit model training; all computation

happens at prediction time. The choice of k and the distance metric significantly affect the model's performance, balancing sensitivity to noise and the ability to capture local patterns in the data [12]

Support Vector Machines

SVMs are supervised learning algorithms used for classification and regression tasks [13]. They aim to find the optimal hyperplane that separates data points of different classes with the maximum margin. The data points closest to this hyperplane are called support vectors, and they are critical in defining the decision boundary. When data is not linearly separable, SVM employs kernel functions to transform the input space into a higher-dimensional feature space where a linear separator may exist. This approach allows SVM to handle complex, non-linear relationships in the data. The choice of kernel and regularization parameters significantly influences the model's performance and generalization ability.

Logistic Regression

Logistic Regression is a supervised learning algorithm used for binary and multi-class classification tasks [14]. It models the probability that an input belongs to a particular class by applying the logistic (sigmoid) function to a linear combination of input features. The model outputs a probability between 0 and 1, which can be thresholded to make a discrete prediction. Logistic Regression is trained by minimizing a loss function, typically the cross-entropy loss, using optimization methods like gradient descent. Although based on a linear decision boundary, it can capture non-linear relationships when combined with feature transformations or interactions.

Random Forest

Random Forest is a supervised ML method that combines multiple decision trees to improve prediction accuracy and robustness [15]. Each tree is built from a random bootstrap sample of the training data, and at each split, a random subset of features is considered. This combination of bagging (bootstrap aggregation) and feature randomness ensures that the trees are diverse and weakly correlated, reducing overfitting compared to individual decision trees. For classification tasks, predictions are made by majority voting among the trees; for regression tasks, the outputs are averaged. Decision trees, the base models, partition the feature space by asking questions that split the data into subsets leading to predictions. Random Forest refines this by aggregating over many such trees, using both bagging and feature subspacing, to achieve better generalization.

2.3.2 Unsupervised Learning Methods

K-Means

K-Means is an unsupervised learning algorithm used to partition data into k distinct clusters based on feature similarity [16]. The process begins by randomly initializing k centroids. Each data point is then assigned to the nearest centroid, forming clusters. Subsequently, the centroids are recalculated as the mean of all points in their respective clusters. This assignment and update process repeats iteratively until

the centroids stabilize, indicating convergence. The algorithm aims to minimize the within-cluster sum of squares, effectively reducing the variance within each cluster. K-Means is computationally efficient and widely used in various applications, such as customer segmentation and image compression. However, it requires the number of clusters k to be specified in advance and may converge to local minima depending on the initial centroid positions.

2.4 Deep Neural Networks

Deep Neural Networks (DNNs) are a class of models within the broader field of ML that consist of multiple layers of artificial neurons. Inspired by biological neural systems, these networks are designed to learn complex, hierarchical representations directly from raw input data. Unlike traditional ML methods that rely on hand-crafted features, DNNs automatically extract high-level abstractions through a series of non-linear transformations, making them particularly effective in tasks such as image recognition, natural language processing, and time series analysis [17].

At the core of a DNN is the artificial neuron, which computes a weighted sum of its inputs, adds a bias, and applies a non-linear activation function. For a given layer l, the output $\mathbf{h}^{(l)}$ is computed as:

$$\mathbf{h}^{(l)} = \phi\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right),$$

where $\mathbf{W}^{(l)}$ is the weight matrix, $\mathbf{b}^{(l)}$ is the bias vector, and $\phi(\cdot)$ represents an activation function such as ReLU or sigmoid. The parameters $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are optimized via backpropagation, which computes gradients of a loss function $\mathcal{L}$ using the chain rule [18].

2.4.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of DNN that are particularly well-suited for processing grid-like data such as images [18]. The key innovation of CNNs is the use of convolutional layers, which apply learnable filters to local regions of the input to extract spatially invariant features. For instance, for a one-dimensional input signal $\mathbf{x} \in \mathbb{R}^T$ and a filter $\mathbf{w} \in \mathbb{R}^k$, the convolution operation is defined as:

$$z_t = \sum_{i=0}^{k-1} w_i \cdot x_{t+i} + b,$$

where b is a bias term and z_t is the output at time t. Multiple filters generate different feature maps, each capturing distinct aspects of the input. Following convolution, pooling layers (e.g., max pooling) are used to downsample these feature maps, reducing computational complexity and helping to ensure the network is robust to small shifts in the input. Finally, fully connected layers integrate the extracted features for the final prediction.

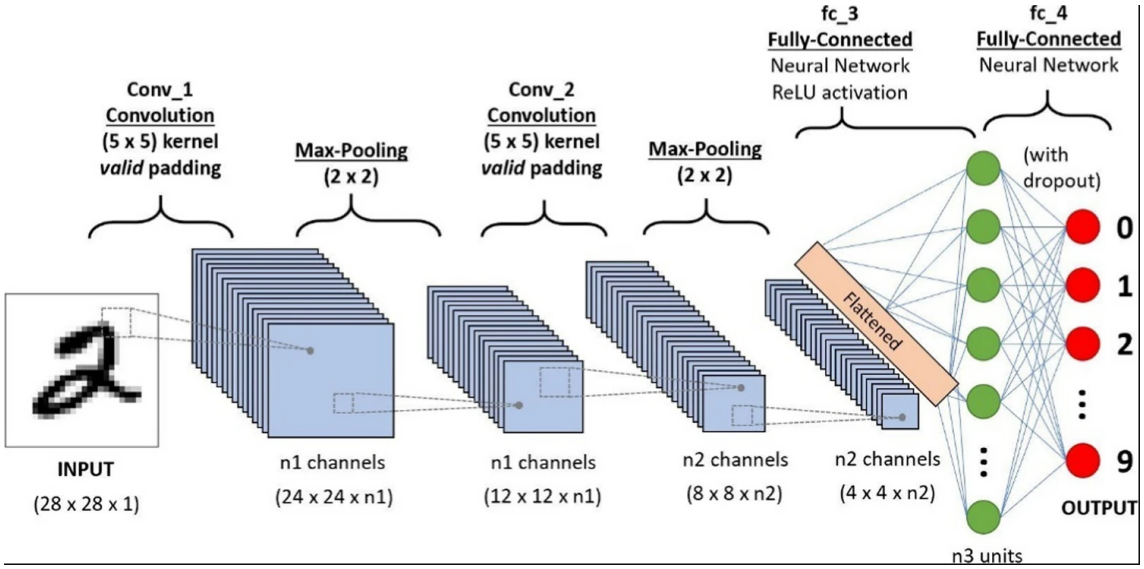


Figure 2.3: Connectivity of the pooling and kernels in CNNs [19]

2.4.2 Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) specifically designed to overcome the limitations of standard RNNs, such as the vanishing gradient problem. LSTMs introduce memory cells and gating mechanisms that allow the network to retain information over long time periods, making them especially useful for sequential data processing [18].

The internal structure of a LSTM cell, with its three gating units and memory update pathway, is shown in Figure 2.4. This diagram highlights how information is selectively forgotten, added, and output at each time step.

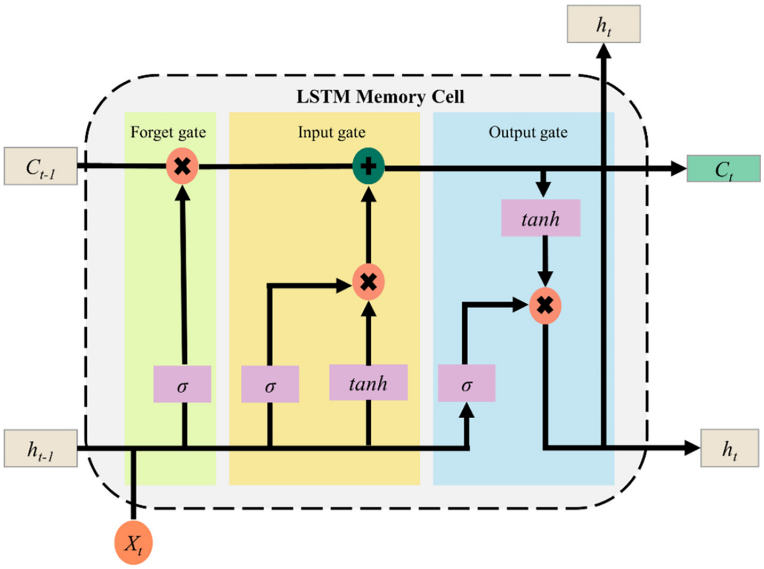


Figure 2.4: Internal connectivity of an LSTM memory cell, illustrating the forget, input, and output gates that regulate the flow of information across time steps [19].

Each LSTM cell is composed of three primary gates:

- **Forget Gate:** Determines which information from the previous cell state $\mathbf{C}_{t-1}$ should be discarded. It is defined as:

$$\mathbf{f}_t = \sigma\Big(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\Big),$$

  where $\sigma(\cdot)$ is the sigmoid function.

- **Input Gate:** Decides which new information should be added to the cell state. It computes an input modulation gate:

$$\mathbf{i}_t = \sigma\Big(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\Big)$$

  and a candidate cell state:

$$\tilde{\mathbf{C}}_t = \tanh\Big(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C\Big).$$

- **Output Gate:** Determines which information from the cell state should be emitted as the hidden state:

$$\mathbf{o}_t = \sigma\Big(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\Big),$$

  and updates the hidden state:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t),$$

  where the new cell state is

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} \; + \; \mathbf{i}_t \odot \tilde{\mathbf{C}}_t.$$

The gating mechanisms in LSTMs allow effective retention of long-term dependencies while mitigating the vanishing gradient, making them powerful for tasks such as language modeling and time series forecasting.

2.4.3 Hybrid CNN–LSTM Architectures

Hybrid CNN–LSTM models combine the spatial feature extraction strength of Convolutional Neural Networks (CNNs) with the temporal dependency modeling power of Long Short-Term Memory (LSTM) networks. A CNN is first applied *frame-wise*: each time step of the input sequence is processed independently but with shared convolutional weights, yielding a compact representation of the spatial (or feature-channel) structure. The resulting sequence of per frame descriptors is then passed to an LSTM that learns dependencies across time. The overall processing flow is shown in Figure 2.5.

Mathematically, let $\mathbf{X} \in \mathbb{R}^{T\times d}$ denote a length T multivariate series whose t-th row $\mathbf{X}_t \in \mathbb{R}^d$ is either (i) a d-dimensional feature vector (for 1-D sensor data) or (ii) a flattened image or tensor slice if the raw modality is visual. A *per frame* CNN acts

only on the spatial (or channel) axes and performs no pooling along the temporal axis, producing

$$\mathbf{Z} \;=\; \left[\mathbf{Z}_1^\top, \ldots, \mathbf{Z}_T^\top\right]^\top \;\in\; \mathbb{R}^{T\times m},$$

where m is the number of convolutional filters and the time length is preserved. The sequence $\{\mathbf{Z}_t\}_{t=1}^{T}$ is then fed into an LSTM to capture temporal dynamics:

$$\mathbf{h}_t = \mathrm{LSTM}\big(\mathbf{Z}_t, \mathbf{h}_{t-1}\big), \qquad t = 1, \ldots, T,$$

with $\mathbf{h}_0$ denoting the initial hidden state. The final hidden state $\mathbf{h}_T$ or the whole sequence $\{\mathbf{h}_t\}$ can subsequently be passed to fully connected layers for classification or regression, depending on the task.

Hybrid CNN–LSTM architectures have shown state-of-the-art performance in a variety of spatiotemporal applications, such as video action recognition, traffic-flow forecasting, and sensor-based activity recognition [20].

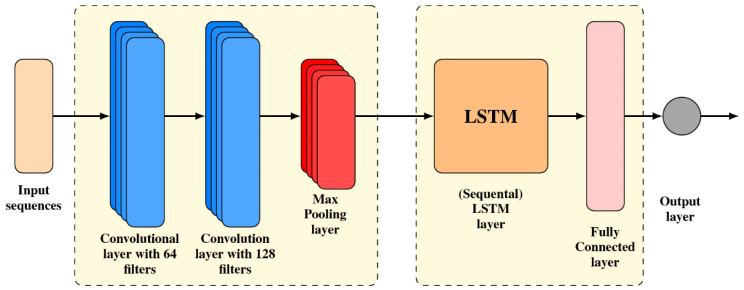


Figure 2.5: Schematic of a hybrid CNN–LSTM architecture: a frame-wise CNN extracts spatial features at each time step, and the resulting feature sequence is processed by an LSTM to model temporal dependencies [21].

2.5 Transfer Learning

Transfer learning utilizes knowledge gained from previously trained models to improve performance on new, related tasks. This approach is especially valuable when working with limited datasets, as it reduces both computational cost and training time [22].

2.5.1 ResNet Architecture

ResNet (Residual Network) was introduced in 2015 as a solution to the degradation problem that occurs when training very deep networks. It achieves this through the use of residual connections, which allow the network to learn identity mappings by bypassing one or more layers [23]. These shortcut connections help preserve gradient flow during backpropagation, mitigating the vanishing gradient issue that typically arises in deep neural networks.

ResNet is available in multiple variants, each differing in depth and parameter count:

Table 2.1: Comparison of ResNet architectures on ImageNet dataset

Model	**Top-1 Accuracy**	**Parameters**
ResNet-50	74.9%	25.6M
ResNet-50V2	76.0%	25.6M
ResNet101	76.4%	44.7M
ResNet101V2	77.2%	44.7M
ResNet152	76.6%	60.4M
ResNet152V2	78.0%	60.4M

Among these, ResNet-50 is one of the most commonly used architectures, comprising 48 convolutional layers, a max pooling layer, and an average pooling layer. Studies comparing Convolutional Neural Network architectures have shown that ResNet-50 often yields higher accuracy in image classification tasks compared to models like VGG16 and VGG19 [24].

2.5.2 Transfer Learning Approaches

Transfer learning is typically implemented using one of two strategies:

1. **Feature Extraction:** In this approach, the final classification layer of a pre-trained network is removed, and the remaining layers are used as a fixed feature extractor. The resulting feature representations can then be passed to traditional classifiers such as SVM or a new Softmax layer tailored to the specific task.
2. **Fine-tuning:** This method involves continuing the training of the pre-trained model on a new dataset. Depending on the similarity between the new and original tasks, fine-tuning may involve adjusting all layers or only the higher ones. Lower layers tend to capture generic features like edges and textures, while deeper layers capture more task-specific representations.

3

Methods

This chapter outlines the methodology employed in this study. It begins with a description of the dataset, including the extraction and preprocessing of radar heatmaps and 3D keypoints. The design of the CNN–LSTM architecture is then presented, followed by the evaluation metrics used to assess model performance. The chapter continues with details on cross-validation strategies and data scaling for different input modalities. It also includes an exploration of unsupervised clustering techniques for 3D keypoint features and introduces a regression-based framework for estimating training data requirements. Finally, a cluster-based risk assessment framework is proposed to support analysis of classification uncertainty.

3.1 Dataset

The dataset consists of recordings from two modalities: radar and camera.[1] For each recording session, a test subject is seated in the front passenger seat of a vehicle in a stationary environment, and both radar and camera data are captured simultaneously. Prior to each session, the subject's basic information (including occupant class, age, height, and weight) is recorded manually by the company. While the dataset includes demographic attributes such as age, height, and weight, it does not directly provide occupant class labels (i.e., adult, child forward-facing, or child rearward-facing). As we describe in Section 3.1.1, these demographic features are used to infer the occupant class during the label generation process.

The radar data has been pre-processed by the company and is available in the form of two-dimensional heatmaps with dimensions of 25 by 34. Each heatmap is generated by processing multiple consecutive radar raw data readings, which are aggregated to form a single heatmap representation. The number of radar frames collected per session varies considerably, with most sessions containing between 200 and 600 frames, several exceeding 700 frames, including one with over 1,100 frames, and a few sessions containing fewer than 100 frames.

The camera system simultaneously captures raw images and extracts derived features such as 2D/3D keypoints and posture information. The keypoints, which include data points such as ears, nose, hips, and shoulders, are extracted using an existing

[1]The dataset used in this thesis is provided by Volvo Cars and is confidential. It will not be made publicly available due to proprietary restrictions.

neural network implementation. The camera data is processed separately and is not fused with the radar modality in the current study. The camera-derived keypoints serve as additional features that may be used in a separate classification pipeline.

The dataset exhibits an imbalanced distribution among the occupant classes. In particular, there are 11 instances labeled as Adult, 38 as Child Forward-Facing, and 12 as Child Rearward-Facing.

3.1.1 Building the Initial Labeled Dataset from ROSBags

1. Extracting Raw Sensor Data
Volvo's dataset was stored in Robot Operating System bag files (ROS bags), which are commonly used to record and replay sensor data in robotic applications. However, the data was not readily available in a structured format, requiring manual extraction and organization.

To achieve this, we processed the ROS bag files to extract relevant sensor data, including camera images, radar heatmaps, occupant classification probabilities, and occupant keypoints. Each data type was saved in appropriate formats (e.g., `.png`, `.npy`, and `.csv`) for further analysis and modeling.

2. Identifying and Removing Noisy Data
During data collection, segments of noisy or corrupted data can appear when a test engineer or parent interacts with the occupant, such as adjusting a child's posture or restraining system through the open door of the stationary vehicle. This type of interference is specific to the controlled testing setup and would not occur under real-world operating conditions, where external human intervention inside the cabin during driving is not expected. To mitigate this, a script was developed to manually annotate and exclude these periods of noise using specified start and end timestamps. The script then:

- Removes all associated image and heatmap data within the specified time range, including both visual outputs and their corresponding metadata.
- Ensures that all related records are deleted from the relevant logs to prevent contamination of downstream analyses or model training.

This helps ensure that external interference data does not propagate into subsequent analyses or model training phases.

3. Linking Occupant Information
A separate CSV file (the search results) provided by Volvo was used to match each ROS bag (by `User ID`) with occupant attributes such as height, weight, and seating direction. A new `Passenger_Id` was assigned to track each occupant. The occupant classification model supplier already designated the labels `4`, `5`, and `7` for different occupant categories (e.g., backward-facing, forward-facing, or adult). Consequently, a `true_class` label was defined based on specific thresholds:

- If the occupant's seating direction is *Backward*, `true_class` = 4.

- If Height $>$ 139.7 cm and Weight $>$ 46.7 kg and the direction is *Forward*, `true_class` = 7.
- If Height $\leq$ 124.5 cm and Weight $\leq$ 25.6 kg and the direction is *Forward*, `true_class` = 5.

These thresholds are based on child safety recommendations outlined in [5].

3.1.2 Preprocessing of Azimuth Radar Heatmaps

Radar heatmaps in their raw form contain highly unbounded intensity values ranging from approximately 14.66 to over 21 million in our dataset. Such extreme magnitudes can impair the performance of many ML algorithms. Traditional models like SVMs, logistic regression, and k-nearest neighbors are particularly sensitive to unscaled features, and even gradient-based methods can suffer from poor convergence or numerical instability.

To address this, two separate preprocessing pipelines were used, depending on the type of model:

For traditional ML models, the following steps were applied:

- **Logarithmic transformation (`log1p`)** was applied to compress the dynamic range of pixel values while preserving relative differences. The `log1p` function is numerically stable and handles zero values gracefully.
- **Standard scaling** (zero mean, unit variance) was then performed using parameters fitted on the training data. This step ensures that each pixel (treated as a feature) contributes equally in models that rely on feature scaling.

After transformation, the pixel values were reduced to a much narrower range (approximately -3.54 to 14.01), improving training stability and model performance.

For deep learning architectures (CNN+LSTMs), the preprocessing followed standard image normalization practices:

- **Min–max normalization to [0,1]** was first applied by scaling each heatmap individually based on its own minimum and maximum values.
- **Rescaling to the range [-1,1]** was then performed to align the data with common expectations of popular CNN architectures (e.g., ResNet), which often assume input in this range.

This dual-track preprocessing strategy ensured that both classical and deep learning models could operate on appropriately scaled and normalized radar inputs, improving convergence behavior and reducing the risk of overfitting to raw intensity patterns.

3.1.3 Preprocessing of 3D Keypoints

ML models, particularly those trained on human pose data, are highly sensitive to the reference frame of keypoints. In our dataset, the 3D keypoints were originally defined relative to a fixed coordinate system located within the vehicle. However,

this fixed reference frame introduced systematic offsets whenever the occupant or seat shifted, even if the occupant's posture remained unchanged. These shifts could lead to misleading model learning, where the model associates seat position with posture, rather than focusing on the occupant's actual pose.

To address this, we preprocessed the 3D keypoints by re-centering all coordinates around the occupant's pelvis. This approach ensured that postures were normalized relative to the occupant, removing unwanted variance caused by seating position or camera alignment.

1. Pelvis Center Calculation

The **pelvis center** was defined as the midpoint between the left and right hip joints. For each frame (or row) in the dataset, the 3D coordinates of the pelvis were computed as follows:

$$\text{pelvis}_x = \frac{\text{left\_hip}_x + \text{right\_hip}_x}{2} \tag{3.1}$$

$$\text{pelvis}_y = \frac{\text{left\_hip}_y + \text{right\_hip}_y}{2} \tag{3.2}$$

$$\text{pelvis}_z = \frac{\text{left\_hip}_z + \text{right\_hip}_z}{2} \tag{3.3}$$

2. Re-Centering Operation

Once the pelvis position was determined, the full 3D pose was re-expressed in a pelvis-centered coordinate system by subtracting the pelvis position from each joint coordinate. Let x_i, y_i, and z_i represent the original coordinates of a joint i, and x_i', y_i', and z_i' represent the normalized coordinates. Then:

$$x_i' = x_i - \text{pelvis}_x \tag{3.4}$$

$$y_i' = y_i - \text{pelvis}_y \tag{3.5}$$

$$z_i' = z_i - \text{pelvis}_z \tag{3.6}$$

This transformation aligns all pose data to a consistent reference point located at the pelvis, which is anatomically central and generally stable, making it an ideal origin for human pose normalization.

3. Note on Quality-Based Filtering:

A quality-based filtering procedure was applied *only* in the context of unsupervised clustering not for the supervised classification experiments. The 3D keypoints were extracted from a camera-based system operating at approximately 10 frames per second, with each keypoint accompanied by a confidence-based quality score. Due to variations in occupant motion and occlusion, some frames exhibited low-quality estimates, which could compromise the reliability of clustering.

To mitigate this, we computed the mean quality score per frame across all 3D keypoints and filtered out frames that fell below the global dataset mean. This thresholding ensured that only frames with sufficiently reliable spatial data were included in the clustering process. The logic for this filtering is illustrated in Figure 3.1, which shows the average 3D keypoint quality per class. As seen in the figure, Class 4 (Child-Rearward) had consistently poor quality scores across most keypoints. As a result, **Class 4 was excluded entirely from the clustering analysis** to avoid introducing unreliable data.

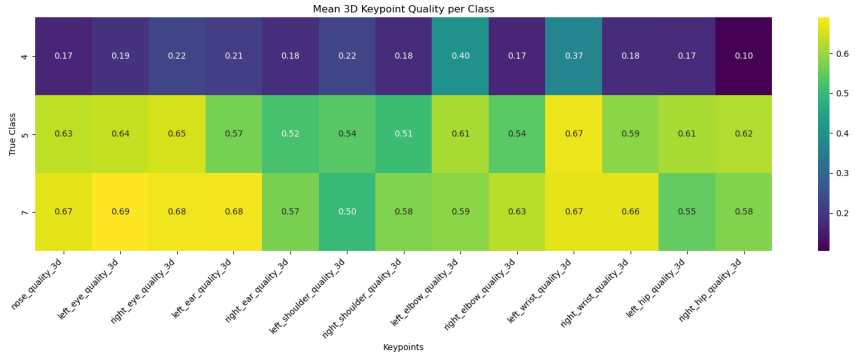


Figure 3.1: **Mean 3D Keypoint Quality per Class.** Each cell shows the average confidence score (ranging from 0 to 1) for a specific keypoint, aggregated by true class label. Class 4 (Child-Rearward) exhibits consistently low quality across nearly all keypoints, motivating its exclusion from unsupervised clustering.

In contrast, all classes were retained for supervised classification tasks. To account for noisy keypoints during multimodal training, we implemented a frame-level *quality-based masking mechanism*. Specifically, if any of the five lowest-scoring keypoints (nose, right wrist, right hip, right elbow, left hip) had a confidence score below 0.50 in a given frame, the corresponding keypoint vector was replaced with zeros and flagged via a binary mask.

These keypoints were selected based on their particularly poor quality in Class 4, which ensures that:

- Class 4 samples are masked more frequently due to their inherent unreliability,
- Extremely low-quality samples in other classes are also masked, avoiding noisy inputs during training,
- The network remains exposed to all classes while being robust to unreliable pose estimates.

This masking strategy allowed the model to ignore corrupted spatial features without excluding valuable radar or temporal data, thus maintaining generalization while reducing the impact of low-confidence inputs.

3.2 Architecture Design

3.2.1 CNN–LSTM

The network consists of two main parts: one for extracting features from each image and another for capturing the sequence of these features over time. Figure 3.2 shows an overview of this design.

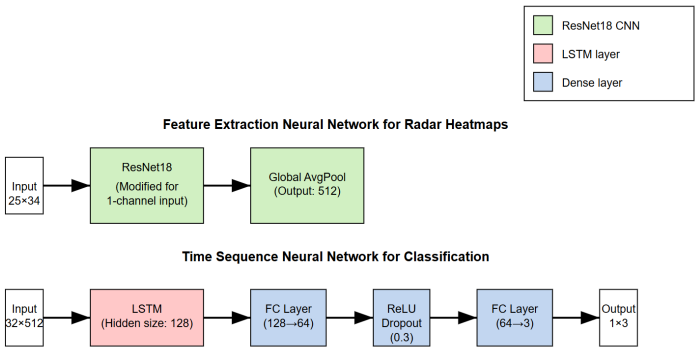


Figure 3.2: CNN–LSTM network architecture

1. **CNN for Feature Extraction:** A modified ResNet-18 is used to process each image. The first convolutional layer is adjusted to accept single-channel input, rather than the standard three channels used for RGB images. All layers of the ResNet-18 are fine-tuned during training, allowing the network to fully adapt to the radar heatmap data. After processing, each image is converted into a 2048-dimensional feature vector. We also conducted experiments using other architectures, including ResNet-50 and a custom CNN.

2. **LSTM for Sequence Processing:** The sequence of feature vectors (sequence length of 32) is passed through a single-layer LSTM with 128 hidden units. Only the output from the final time step of the LSTM is used, summarizing the entire input sequence into a single 128-dimensional vector.

3. **Final Classification:** The 128-dimensional vector from the LSTM is first passed through a fully connected layer to reduce its size to 64. A ReLU activation function is applied, followed by a dropout layer with a probability of 0.3 to prevent overfitting. Finally, another fully connected layer maps the features into three output scores corresponding to the three target classes.

3.3 Evaluation Methods

To assess the effectiveness and performance of ML models on unseen data, several standard evaluation metrics are employed. This section outlines the key metrics implemented in this thesis to evaluate classification performance.

3.3.1 Confusion Matrix

The confusion matrix [25] serves as a fundamental evaluation tool in ML, applicable to both binary and multi-class classification tasks. It provides a tabular representation of a model's predictions compared to the actual outcomes, revealing patterns of correct classifications and misclassification.

The matrix consists of four primary components in binary classification:

1. **True Positive (TP):** Instances where the model correctly identifies the positive class as positive.
2. **True Negative (TN):** Instances where the model correctly identifies the negative class as negative.
3. **False Positive (FP):** Instances where the model incorrectly classifies the negative class as positive (Type I error).
4. **False Negative (FN):** Instances where the model incorrectly classifies the positive class as negative (Type II error).

In multi-class scenarios, the confusion matrix expands to accommodate multiple classes, with the diagonal elements representing correct classifications for each class, while off-diagonal elements represent misclassification.

3.3.2 Accuracy

Accuracy [26] represents one of the most straightforward evaluation metrics, measuring the overall correctness of a model's predictions. It is calculated as the ratio of correct predictions to the total number of predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

While intuitive and widely used, accuracy can be misleading when dealing with imbalanced datasets. For instance, in a dataset where 95% of samples belong to one class, a model could achieve high accuracy by simply predicting the majority class in all cases without actually learning meaningful patterns.

3.3.3 Precision

Precision [26] focuses on the positive predictions made by the model and measures what fraction of these predictions were actually correct. It is particularly valuable in contexts where false positives carry significant costs:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.8}$$

This metric answers the question: "Of all instances predicted as positive, what proportion was actually positive?"

3.3.4 Recall

Recall [26] (also known as sensitivity or true positive rate) measures the model's ability to identify all relevant instances of the positive class. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.9}$$

Recall addresses the question: "Of all actual positive instances, what proportion was correctly identified?" This metric is especially important in scenarios where missing a positive instance (false negative) carries high costs, such as in medical diagnostics or security applications.

3.3.5 F1-Score

The F1-score [26] combines precision and recall into a single metric, representing their harmonic mean:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.10}$$

Ranging from 0 to 1 (with 1 being optimal), the F1-score is particularly useful for imbalanced datasets where a balance between precision and recall is desired. The harmonic mean ensures that the F1-score will be closer to the lower of the two values, requiring both precision and recall to be reasonably high for a good F1-score.

3.3.6 Macro-Averaging for Multi-Class Classification

For multi-class classification problems, the binary metrics (precision, recall, F1-score) can be extended using macro-averaging techniques. This approach calculates the metric independently for each class and then takes the average across all classes [26].

For a classification problem with K classes, the macro-averaged metrics are computed as follows:

$$\text{Macro-average Precision} = \frac{\sum_{k=1}^{K} \text{Precision}_k}{K}$$

$$\text{Macro-average Recall} = \frac{\sum_{k=1}^{K} \text{Recall}_k}{K}$$

$$\text{Macro-average F1-score} = \frac{\sum_{k=1}^{K} \text{F1-score}_k}{K}$$

A key characteristic of macro-averaging is that it treats all classes equally, regardless of their frequency in the dataset. This provides a balanced assessment across all classes but may understate overall performance if rare classes perform poorly compared to

common ones. Macro-averaging is particularly useful when evaluating models on datasets with class imbalance, as it prevents the metrics from being dominated by the majority class.

3.4 Cross-Validation Setup and Data Scaling

This section presents the cross-validation performance of several model families, including CNN–LSTM architectures applied to radar heatmaps and supervised models trained on both heatmaps and 3D keypoint-derived features. All models except for the keypoint-based supervised classification, which was trained on a single data split rather than full cross-validation were trained using the stratified data splits and training ratios detailed in Section 4.2.1.

3.4.1 CNN–LSTM on Radar Heatmaps

We evaluate three CNN–LSTM architectures: two using ResNet backbones (ResNet-18 and ResNet-50), and one using a custom TinyCNN. The TinyCNN is a lightweight convolutional network specifically designed for small radar heatmaps (25×34), featuring three convolutional blocks followed by an adaptive average pooling layer. All models are trained on azimuth heatmaps extracted from radar sequences, capturing spatiotemporal dynamics via an LSTM layer.

Each setup is evaluated using overall classification accuracy and per-class F1-scores, providing a detailed view of model behavior across different architectures and data scales.

3.4.2 ResNet + LSTM (All 3D Keypoints)

This model combines radar heatmap encodings with full 3D body keypoints (e.g., nose, eyes, shoulders, hips), which are preprocessed as described in Section 3.1.3. The ResNet processes each radar frame into spatial features, and the LSTM aggregates these across time. Keypoint vectors are also aggregated over the subsequence and fused with the LSTM output before final classification.

3.4.3 ResNet + LSTM (3D Body Metrics)

This variation follows the same preprocessing steps described in Section 3.1.3, but instead of using full keypoint arrays, we compute a reduced set of derived body measurements (e.g., hip width, shoulder width, torso length) as input features. These metrics are extracted from each frame, and their per-window averages are concatenated with the LSTM output.

3.4.4 Supervised Models on Heatmaps / Keypoints

In addition to deep learning models, we evaluate a set of traditional supervised classifiers including SVM, XGBoost, and Logistic Regression on both radar heatmaps

and 3D keypoint data. These approaches offer interpretable baselines and help assess the discriminative power of raw radar and body-pose features when used independently.

Heatmap-Based Classification

In this setting, each 25×34 azimuth heatmap is reshaped into a flat 850-dimensional feature vector. For each subsequence window of 32 frames (with a stride of 10), we compute the mean heatmap across all frames in the window. This averaged heatmap is then flattened to produce a single 850-dimensional feature vector per window. Preprocessing details for the heatmaps are provided in Section 3.1.2.

3D Keypoint-Based Classification Classes: Child-Forward and Adult

We also evaluate classical supervised models on 3D keypoints for binary classification between Child-Forward and Adult classes. The Child-Rearward class is excluded due to occlusion: the camera cannot reliably detect keypoints for rear-facing children, as their bodies are obstructed by the seat.

Each 32-frame subsequence is represented by a single feature vector, obtained by averaging all 3D keypoints across time. To account for pose variability, we preprocess the data as described in Section 3.1.3. All frames are retained during training; no filtering based on keypoint confidence was applied.

- **Validation Protocol:** A fixed train/test split was used, with no cross-validation folds.
- **Training Set Size:** The training size was fixed; no scaling experiments were conducted.
- **Feature Importance:** After training, we performed feature importance analysis to identify which keypoints most influenced the classification outcome.

3.5 Unsupervised Clustering of 3D Keypoint Features

In addition to supervised learning, we investigated the underlying structure of the 3D keypoint data through unsupervised clustering. The goal was to determine whether keypoint-based body metrics could reveal natural groupings among occupants (e.g., Adult vs. Child-Forward).

As explained in Section 3.1.3, low-quality data were filtered based on keypoint confidence scores to reduce the impact of noisy estimates. Additionally, Class Child-Rearward was excluded from this analysis due to consistently unreliable keypoint estimates in that group.

3.6 Estimating Training Data Requirements via Performance Extrapolation

To investigate how much additional data is required to achieve desired classification performance in radar-based occupant detection, we adopt and extend the methodology from Mahmood et al. [27]. Their work systematically evaluates regression-based extrapolation models to estimate dataset requirements for downstream tasks, providing both theoretical insight and practical simulation tools for efficient data planning.

3.6.1 Regression-Based Estimation Framework

Following the approach outlined by Mahmood et al., we model the relationship between training dataset size and classification performance using concave, monotonically increasing functions. Specifically, we fit four nonlinear regression functions:

- **Power Law:** $\hat{v}(n;\theta) = \theta_1 n^{\theta_2} + \theta_3$
- **Arctangent:** $\hat{v}(n;\theta) = \frac{200}{\pi} \arctan\left(\frac{\theta_1}{\pi/2} n + \theta_2\right) + \theta_3$
- **Logarithmic:** $\hat{v}(n;\theta) = \theta_1 \log(n + \theta_2) + \theta_3$
- **Root (Algebraic):** $\hat{v}(n;\theta) = \frac{100n}{(1+|\theta_1 n|^{\theta_2})^{1/\theta_2}} + \theta_3$

These models take training data proportion as input and predict a scalar performance metric, such as F1-score. The fitting is performed independently for each classification target, and higher training proportions are withheld during training to simulate an out-of-sample prediction scenario.

This framework is classifier-agnostic and can be applied to any supervised model, traditional or deep, as long as performance metrics are available across varying training sizes.

3.6.2 Performance Targeting and Error Analysis

To evaluate extrapolation effectiveness, we compare predicted training requirements for reaching specific performance thresholds against the actual data requirements observed empirically. This comparison allows us to compute:

- The coefficient of determination (R^2) for model fit quality
- An efficiency ratio, defined as the ratio between predicted and actual additional data needed to meet performance targets

An efficiency ratio close to 1.0 indicates accurate prediction; values greater than 1.0 imply overestimation, while values less than 1.0 indicate underestimation. These metrics provide practical insights into each model's forecasting behavior, especially near learning curve saturation points.

3.6.3 Motivation from Prior Work

This methodology is rooted in the findings of Mahmood et al. [27], who demonstrated that while traditional scaling laws (e.g., Power Law) offer useful performance extrapolation, they can exhibit significant variance in real-world data collection scenarios. They highlight the risk of relying solely on R^2 for extrapolation accuracy and advocate for using ensemble estimators, correction factors, and multiple collection rounds to improve reliability. Our work builds on their regression strategies but adapts them specifically to the context of radar-based occupant classification with class-wise performance evaluation.

3.7 Cluster-Based Risk Assessment Framework

While the evaluation metrics outlined earlier provide essential insights into model performance, they offer limited guidance on why specific misclassifications occur or which training sequences hinder generalization. To address this limitation, we present a cluster-based approach that analyzes training sequence difficulty and links validation errors to specific regions of the training data. This data-centric method enables targeted interventions, such as risk-informed data augmentation, as described below.

3.7.1 Sequence-Level Risk Analysis Workflow

To identify training sequences that may negatively impact generalization, we combine unsupervised clustering, training loss analysis, and validation error attribution into a unified risk assessment strategy.

1. **Clustering Training Sequences:** After training the model, we extract feature vectors from an intermediate layer (CNN+LSTM) for each training sequence. We then apply an unsupervised clustering algorithm (KMeans) to group sequences with similar representations. Each sequence is assigned a cluster ID based on its embedding.

2. **Tracking Training Loss:** During training, we log the loss for each sequence. These per-sample losses reflect the difficulty the model had in learning those examples. For each cluster, we compute the average training loss, providing a measure of how consistently challenging the cluster was for the model.

3. **Linking Validation Errors to Clusters:** After training, we analyze validation misclassifications. For each misclassified validation sequence, we compute its feature embedding and identify the most similar training sequence via cosine similarity. The validation error is then attributed to the cluster of the closest training sample. We count how many validation errors are linked to each cluster.

3.7.2 Risk Label Assignment

With each cluster characterized by its average training loss, number of training samples, and number of linked validation errors, we assign a risk label using a percentile-based thresholding strategy. Specifically, clusters are labeled as follows:

- **Stable**: No validation errors linked to the cluster.
- **Medium-risk**: Cluster has a high average training loss or a moderately high error ratio.
- **High-error-only**: Cluster has a high validation error ratio but low training loss.
- **Review**: Cluster has both a high average training loss and a high validation error ratio.

These risk labels guide targeted data augmentation. Medium-risk clusters are selected for oversampling to improve robustness on challenging yet learnable patterns.

4

Results

This chapter presents the experimental results obtained throughout the study. It begins with an evaluation of the baseline model currently deployed by Volvo Cars. The performance of various CNN–LSTM architectures on radar heatmap classification is then detailed, followed by results from multimodal experiments combining radar data with 3D keypoints. Further sections report on supervised model performance, class-wise metrics, and targeted classification for specific action classes. The chapter also includes clustering analyses of 3D keypoints, an assessment of training data requirements using regression-based estimation, and concludes with an evaluation of how a cluster-based risk assessment framework influences model performance and reliability.

4.1 Evaluation of the Deployed Volvo Cars Model

In this section, we present the performance of the occupant-classification model, which targets three primary classes:

- Class 4: Backward-facing occupant
- Class 5: Forward-facing child occupant
- Class 7: Adult occupant

It is important to note that this model was trained using data collected over a limited period (approximately 1.5 weeks) and lacked representation of certain occupant classes, most notably backward-facing children. As a result, the model does not reflect the capabilities of more recent systems and is included here solely for baseline comparison.

The model was trained on azimuth heatmaps generated from radar signals. As summarized in Table 4.1 and visualized in Figure 4.1, it exhibits poor overall accuracy and especially struggles with Class 4 detection.

Table 4.1: Classification Report for the Occupant-Classification Model

Class	Precision	Recall	F1-score	Support
Child-Rearward	0.89	0.01	0.01	2920
Child-Forward	0.68	0.44	0.54	17545
Adult	0.24	0.66	0.35	5299
Overall Acc.	0.44 (on 25,764 samples)			

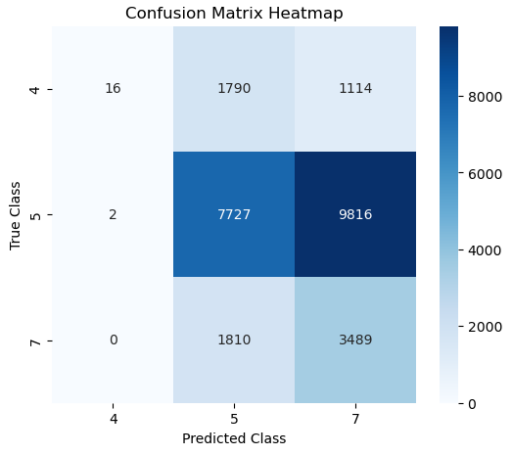


Figure 4.1: Confusion Matrix for the Occupant Classification Deployed Model.

Key Observations.

- **Backward-facing:** Extremely low recall (≈ 0.01) yields a near-zero F1-score, indicating almost no correct detections for backward-facing occupants.
- **Forward-facing child:** While its precision is higher (0.68), the recall (0.44) is still relatively low, suggesting notable confusion with other classes.
- **Adult:** Despite a higher recall (0.66), poor precision (0.24) indicates frequent misclassification of other classes as adults.

Overall, the model's accuracy of 44% reflects the limitations of the training dataset and the early development stage of this classifier. Its inclusion serves to illustrate the improvements achieved in subsequent, more comprehensive models presented later in this thesis.

4.2 Unified Training Framework

To ensure fair and consistent comparison across modeling approaches, we applied a standardized training protocol across all experiments, including deep learning models (CNN–LSTM), traditional supervised classifiers, and the cluster-based risk assessment study. All models were trained and evaluated using the same data splits, preprocessing steps, and experimental structure.

4.2.1 Data Splitting and Training Ratios

To support robust performance evaluation across different training sizes and folds, we implemented a stratified and class-aware data splitting strategy. This procedure was applied consistently across all experiments, including traditional classifiers, CNN–LSTM models, and cluster-based risk assessment.

We used five-fold cross-validation, where each fold iteration held out one part of the data for validation while training on the remaining four. For each fold, training subsets were generated at five different sizes: 15%, 30%, 50%, 75%, and 100%. This allowed us to examine performance scalability with increasing data availability.

Since data samples are grouped by `Passenger_Id`, splits were created at the passenger level to avoid sample leakage between training and validation. We focused on three primary classes *Child-Rearward*, *Child-Forward*, and *Adult* which were balanced across folds. While the dataset is inherently imbalanced, the class distributions were preserved across all folds, ensuring that the same degree of imbalance was maintained in each split.

For each target class, passengers were distributed into five stratified folds. Fold sizes were equalized as much as possible. In each iteration, the selected fold served as the validation set while the remaining folds comprised the training candidates. Care was taken to ensure no passenger overlapped between training and validation, eliminating data leakage.

From the training pool, a subset of passengers was sampled based on the specified training ratio. Sampling was stratified by class to preserve class distribution. To ensure representativeness, any passengers belonging to multiple classes were also included. Additional checks ensured that all target classes were represented in both training and validation sets, even at low data ratios.

For the *Child-Rearward* class, we also handled subtypes (*infant* and *3yo*) to prevent cross-subtype contamination between train and validation sets. If only one subtype was present in training, validation data from the alternate subtype were excluded. Conversely, if subtype-based exclusion caused missing validation classes, targeted passenger restoration was performed to preserve class presence.

This meticulous process ensured that each fold and ratio maintained balanced class representation, avoided leakage, and respected subtype boundaries. As a result, all models were trained on consistent, reliable splits across experiments, enabling fair performance comparisons.

4.2.2 Radar-Only vs. Multimodal (Radar + Keypoints) Models

Both radar-only and multimodal (Radar + 3D Keypoints) CNN–LSTM models followed the same training pipeline:

- Input sequences were processed into fixed-length windows (32 frames, stride of 10).

- A CNN extracted spatial features, followed by an LSTM to capture temporal dependencies.
- Models were trained using the Adam optimizer with a learning rate scheduler.
- Class-weighted cross-entropy loss was used to address class imbalance.
- Early stopping was applied based on validation macro F1-score.

All CNN–LSTM models were trained across the same five training ratios and cross-validation folds to enable consistent comparison with traditional models.

4.2.3 Traditional Supervised Classifiers

For classical models such as SVM, Random Forest, and XGBoost, radar heatmap sequences were summarized using mean aggregation over time. The resulting feature vectors were:

- Log-transformed and standardized using `StandardScaler`.
- Reduced to 50 dimensions via PCA.
- Used to train each classifier on the same class-balanced folds and training ratios.

Performance was evaluated using macro F1-score and detailed per-class classification metrics, ensuring consistency with the deep learning evaluations.

4.2.4 Cluster-Based Risk Assessment

The same experimental pipeline was used for cluster-based risk analysis. Each model configuration (fold × training ratio × model) was trained **three times** independently to account for variability in initialization and training dynamics. This repeated training enabled robust estimation of performance variance and supported downstream analyses of cluster-level error distributions and model reliability.

4.3 Radar Heatmap Classification with CNN–LSTM Architectures

We evaluated the classification performance of three CNN–LSTM architectures **ResNet-18 + LSTM, ResNet-50 + LSTM, and TinyCNN + LSTM** on radar heatmaps across five training sizes (15%, 30%, 50%, 75%, and 100%). Each model was tested using a five-fold cross-validation strategy, employing a passenger-based split to avoid identity leakage between training and validation sets.

4.3.1 Comparative Model Performance

The average per-class F1-scores across training sizes for all CNN–LSTM architectures are summarized in Table 4.2. Overall, all models exhibit improved performance

as the training size increases, with **Child-Forward** consistently achieving the highest F1-scores, even with limited training data. This trend aligns with the class distribution imbalance discussed in Section 3.1, where Child-Forward has the highest representation in the dataset.

Despite architectural differences, the trends across all models are notably similar:

- Performance variability is higher at smaller training sizes and stabilizes as more data becomes available.
- **ResNet-50 + LSTM** slightly outperforms the other models at larger training sizes, particularly in terms of macro F1.
- **TinyCNN + LSTM**, while compact in size (0.14MB), demonstrates competitive results, especially at 100% training, making it a viable option for deployment in resource-constrained environments.

Table 4.2 shows the average per-class F1-scores (mean ± std) across training sizes for all three models. The trend indicates consistent gains across all classes with more training data, with ResNet-50 + LSTM and TinyCNN + LSTM showing slightly stronger results for the Adult and Child-Rearward classes at 100% training. **Bolded values indicate the best-performing model for each class and training size**.

Table 4.2: Average Per-Class F1-scores Across Training Sizes and Models (Mean ± Std)

Training Size	Class	ResNet-18 + LSTM	ResNet-50 + LSTM	TinyCNN + LSTM
15%	Adult	0.533 ± 0.108	0.457 ± 0.216	$\mathbf{0.555 \pm 0.193}$
	Child-Forward	0.821 ± 0.061	$\mathbf{0.836 \pm 0.040}$	0.817 ± 0.024
	Child-Rearward	$\mathbf{0.598 \pm 0.139}$	0.551 ± 0.125	0.474 ± 0.259
30%	Adult	0.613 ± 0.085	0.655 ± 0.107	$\mathbf{0.665 \pm 0.157}$
	Child-Forward	$\mathbf{0.868 \pm 0.031}$	0.862 ± 0.040	0.866 ± 0.034
	Child-Rearward	0.621 ± 0.206	$\mathbf{0.659 \pm 0.142}$	0.635 ± 0.107
50%	Adult	0.691 ± 0.137	$\mathbf{0.735 \pm 0.127}$	0.702 ± 0.147
	Child-Forward	0.904 ± 0.019	0.897 ± 0.014	$\mathbf{0.906 \pm 0.030}$
	Child-Rearward	$\mathbf{0.748 \pm 0.132}$	0.662 ± 0.149	0.631 ± 0.219
75%	Adult	0.775 ± 0.128	$\mathbf{0.809 \pm 0.140}$	0.736 ± 0.081
	Child-Forward	0.922 ± 0.031	$\mathbf{0.926 \pm 0.037}$	0.896 ± 0.033
	Child-Rearward	0.766 ± 0.132	$\mathbf{0.772 \pm 0.090}$	0.730 ± 0.113
100%	Adult	$\mathbf{0.853 \pm 0.059}$	0.836 ± 0.085	0.775 ± 0.134
	Child-Forward	$\mathbf{0.939 \pm 0.026}$	0.934 ± 0.024	0.922 ± 0.032
	Child-Rearward	0.737 ± 0.221	$\mathbf{0.831 \pm 0.103}$	0.787 ± 0.036

Table 4.3 reports macro F1-score and accuracy across training sizes for each model. While all models demonstrated solid performance, **ResNet-50 + LSTM consistently achieved the highest scores**, especially at full training size. **Bolded values indicate the best-performing model per row and metric.** TinyCNN + LSTM remained competitive, making it a strong candidate for resource-constrained environments.

Table 4.3: Macro F1-socre and Accuracy Across Training Sizes for Each Model (Mean ± Std)

Training Size	ResNet-18 + LSTM	ResNet-50 + LSTM	TinyCNN + LSTM
15%	**0.6506 ± 0.0574** / 0.7345 ± 0.0591	0.6147 ± 0.0680 / **0.7369 ± 0.0236**	0.6153 ± 0.0971 / 0.7113 ± 0.0295
30%	0.7007 ± 0.0841 / 0.7676 ± 0.0351	**0.7254 ± 0.0596** / **0.8056 ± 0.0305**	0.7221 ± 0.0685 / 0.7567 ± 0.0643
50%	**0.7810 ± 0.0531** / **0.8406 ± 0.0240**	0.7647 ± 0.0646 / 0.8269 ± 0.0171	0.7461 ± 0.1099 / 0.8202 ± 0.0548
75%	0.8211 ± 0.0824 / 0.8580 ± 0.0193	**0.8358 ± 0.0719** / **0.8685 ± 0.0416**	0.7872 ± 0.0607 / 0.7838 ± 0.0920
100%	0.8429 ± 0.0808 / **0.8990 ± 0.0314**	**0.8670 ± 0.0462** / 0.8919 ± 0.0279	0.8278 ± 0.0569 / 0.8615 ± 0.0333

4.4 Multimodal CNN–LSTM Experiments

This section presents the classification results for two multimodal CNN–LSTM architectures that combine radar heatmaps with pose-based features. One model integrates full 3D keypoints, while the other uses reduced 3D body metrics (hip width, shoulder width, torso length). Both configurations were evaluated using five-fold cross-validation over training sizes of 15%, 30%, 50%, 75%, and 100%, identical to the radar-only setup.

4.4.1 Comparative Model Performance

Table 4.4 reports the average per-class F1-scores across training sizes. Consistent with radar-only results, the **Child-Forward** class continues to exhibit the highest performance due to its stronger representation. Notably, the 3D keypoints model achieves the highest macro F1-score at full training size, while the body metrics model demonstrates more consistent performance at mid-range data sizes. **Bolded values indicate the higher-performing model for each class and training size**.

Table 4.4: Average Per-Class F1-scores Across Training Sizes for Multimodal Models (Mean ± Std)

Training Size	Class	3D Keypoints	3D Body Metrics
15%	Adult	**0.682 ± 0.181**	0.614 ± 0.157
	Child-Forward	0.854 ± 0.077	**0.857 ± 0.046**
	Child-Rearward	**0.507 ± 0.198**	0.498 ± 0.213
30%	Adult	**0.766 ± 0.125**	0.702 ± 0.131
	Child-Forward	**0.901 ± 0.018**	0.882 ± 0.035
	Child-Rearward	**0.663 ± 0.149**	0.616 ± 0.149
50%	Adult	**0.817 ± 0.096**	0.725 ± 0.177
	Child-Forward	**0.916 ± 0.021**	0.903 ± 0.024
	Child-Rearward	0.653 ± 0.209	**0.720 ± 0.152**
75%	Adult	**0.835 ± 0.082**	0.805 ± 0.120
	Child-Forward	**0.930 ± 0.015**	0.926 ± 0.013
	Child-Rearward	0.721 ± 0.157	**0.787 ± 0.080**
100%	Adult	**0.893 ± 0.066**	0.887 ± 0.096
	Child-Forward	**0.952 ± 0.012**	0.947 ± 0.019
	Child-Rearward	**0.792 ± 0.151**	0.729 ± 0.176

Table 4.5 shows Macro F1-socre and accuracy values across training sizes. The full keypoint model slightly outperforms the body metric model at 100% training, but

the body metric model achieves stronger or comparable results at intermediate levels (50–75%). **Bolded values indicate the higher-performing model (per row, per metric)**.

Table 4.5: Macro F1-score and Accuracy Across Training Sizes for Multimodal Models (Mean ± Std).

Training Size	3D Keypoints	3D Body Metrics
15%	**0.6811 ± 0.1088** / 0.7531 ± 0.0842	0.6559 ± 0.1035 / **0.7647 ± 0.0539**
30%	**0.7766 ± 0.0751** / **0.8224 ± 0.0456**	0.7332 ± 0.0848 / 0.7853 ± 0.0694
50%	**0.7956 ± 0.0838** / **0.8674 ± 0.0305**	0.7827 ± 0.0878 / 0.7747 ± 0.1524
75%	0.8286 ± 0.0696 / 0.8607 ± 0.0207	**0.8393 ± 0.0510** / **0.8726 ± 0.0275**
100%	**0.8794 ± 0.0634** / **0.9080 ± 0.0216**	0.8546 ± 0.0823 / 0.8971 ± 0.0430

4.5 Supervised Models on Heatmaps

This section presents the performance of five supervised classifiers trained on flattened azimuth radar heatmaps, using the experimental setup defined in Section 3.4.4. The evaluated models include Logistic Regression, SVM, Random Forest, k-Nearest Neighbors (k-NN), and XGBoost.

4.5.1 Comparative Model Performance

TTable 4.6 reports the macro-averaged F1-scores (mean ± standard deviation) across five cross-validation folds for each model and training size. Results are shown for training set proportions of 15%, 30%, 50%, 75%, and 100%. **Bolded values indicate the highest macro F1-score achieved at each training size**.

Table 4.6: Cross-Validation Summary for Supervised Models (Heatmap Subsequences, Outliers Included). Scores represent Macro F1-score (mean ± std) across five folds.

	Training Size				
Model	**15%**	**30%**	**50%**	**75%**	**100%**
Logistic Regression	0.6159 ± 0.0923	**0.7513 ± 0.0669**	**0.8131 ± 0.0619**	0.8201 ± 0.0318	0.8027 ± 0.0432
SVM	**0.6221 ± 0.1358**	0.6977 ± 0.1188	0.8043 ± 0.1143	**0.8532 ± 0.1059**	**0.8629 ± 0.1015**
Random Forest	0.4529 ± 0.0993	0.5623 ± 0.0855	0.5731 ± 0.1511	0.6468 ± 0.1330	0.7008 ± 0.0950
k-NN	0.5860 ± 0.1407	0.6759 ± 0.1393	0.7075 ± 0.1343	0.7833 ± 0.1162	0.8108 ± 0.0999
XGBoost	0.5227 ± 0.1089	0.6291 ± 0.0954	0.6901 ± 0.1150	0.7727 ± 0.0930	0.7922 ± 0.0993

Table 4.7 presents the average F1-score for each occupant class (Adult, Child-Forward, Child-Rearward) across five supervised models Logistic Regression, SVM, Random Forest, k-NN, and XGBoost evaluated over five training sizes. Scores are aggregated across five folds per training size. **Bolded values indicate the highest F1-score for each class at a given training size**.

Table 4.7: Average Per-Class F1-scores Across Training Sizes for Supervised Models (Heatmap Subsequences)

Training Size	Class	Logistic Regression	SVM	Random Forest	k-NN	XGBoost
15%	Adult	0.5811	**0.5900**	0.3284	0.4493	0.4144
	Child-Forward	**0.9075**	0.9017	0.8674	0.8750	0.8794
	Child-Rearward	0.3590	0.3747	0.1630	**0.4336**	0.2744
30%	Adult	**0.7781**	0.7652	0.6000	0.6845	0.6825
	Child-Forward	**0.9218**	0.9211	0.8848	0.8993	0.9103
	Child-Rearward	**0.5539**	0.4070	0.2022	0.4439	0.2945
50%	Adult	**0.8432**	0.8354	0.5336	0.6661	0.6924
	Child-Forward	0.9339	**0.9416**	0.8779	0.9059	0.9032
	Child-Rearward	**0.6622**	0.6359	0.3079	0.5504	0.4748
75%	Adult	0.8436	**0.8826**	0.7095	0.7869	0.8070
	Child-Forward	0.9341	**0.9558**	0.9041	0.9310	0.9371
	Child-Rearward	0.6825	**0.7211**	0.3268	0.6320	0.5741
100%	Adult	0.8302	**0.9092**	0.7748	0.8274	0.8516
	Child-Forward	0.9305	**0.9601**	0.9146	0.9439	0.9398
	Child-Rearward	0.6475	**0.7195**	0.4130	0.6611	0.5852

4.6 3D Keypoint-Based Subsequence Classification Child-Forward and Adult

This section presents the performance of classical ML models trained to classify between Child-Forward and Adult occupants using sequence-averaged 3D keypoints, as described in Section 3.4.4. The Child-Rearward class was excluded due to poor keypoint visibility caused by seat occlusion.

Five classifiers were evaluated: Logistic Regression, SVM (RBF kernel), Random Forest, XGBoost, and CatBoost. Final evaluation was conducted on a held-out test set after hyperparameter optimization using grid search.

4.6.1 Comparative Model Performance

Table 4.8 reports the test set performance for each model, including the macro F1-score and per-class F1-scores for Child-Forward and Adult. **Bolded values indicate the highest performance for each metric.**

Table 4.8: Models Performance: Macro and Per-Class F1-Scores

Model	F1 (Macro)	F1 Child-Forward	F1 Adult
Logistic Regression	**0.9744**	**0.9863**	**0.9626**
SVM	0.9533	0.9765	0.9300
Random Forest	0.9202	0.9621	0.8783
XGBoost	0.8616	0.9364	0.7869
CatBoost	0.9533	0.9765	0.9300

4.6.2 Feature Importance Analysis

Feature importance was computed for all models that support this functionality. Figure 4.2 displays the top-15 most influential features across Logistic Regression, Random Forest, XGBoost, and CatBoost. The SVM model is excluded due to its lack of built-in feature importance scores.

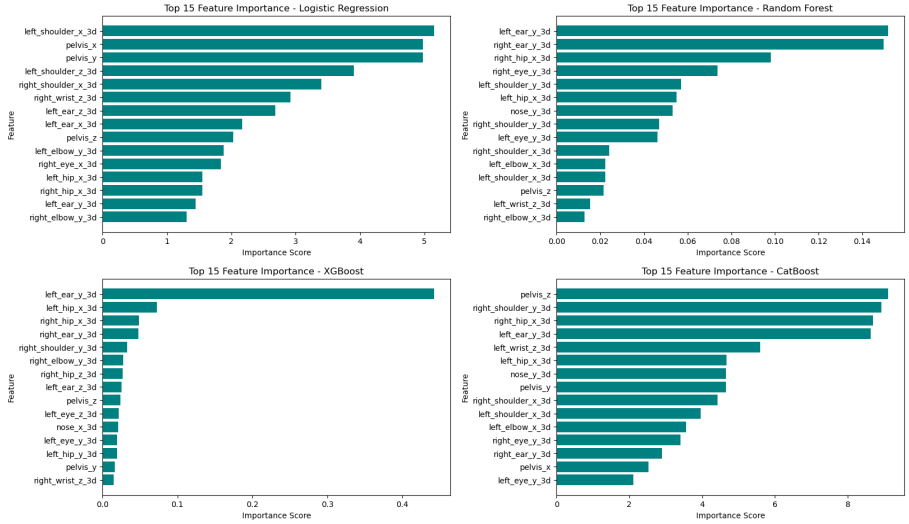


Figure 4.2: Top-15 Most Influential 3D Keypoint Features for Classification Models. Each plot shows the normalized importance scores across models that support feature attribution.

Across models, features from the pelvis, shoulders, and ears particularly in the Y and Z axes appear prominently. For example, pelvis_z and left_ear_y_3d consistently rank high in importance, suggesting that vertical body orientation and head positioning are key for occupant classification. This aligns with the domain intuition that seating posture and upper-body tilt help distinguish between occupant types (e.g., child vs. adult).

Differences in attribution also emerge across models. XGBoost, for instance, places disproportionate importance on a single feature (left_ear_y_3d), indicating potential sensitivity to strong univariate signals. In contrast, CatBoost and Random Forest distribute their focus across a broader range of body joints and axes, implying more complex, multivariate decision boundaries.

These insights not only improve model interpretability but also offer practical value: they can inform feature selection, reduce sensor redundancy, or even optimize sensor placement in future deployments by focusing on the most informative keypoints.

4.7 Clustering Analysis of 3D Keypoints

We investigated the structure of 3D keypoint data using unsupervised clustering techniques to explore natural groupings between Adult and Child-Forward occupants.

As described in Section 3.1.3, low-confidence detections were filtered out, and Class 4 (Child-Rearward) was excluded due to unreliable keypoint estimates.

4.7.1 Body Metrics: Torso, Shoulder, and Hip

Three body geometry features were derived from normalized 3D keypoints:

- **Torso Length:** Distance between the neck and pelvis center
- **Shoulder Width:** Distance between left and right shoulder joints
- **Hip Width:** Distance between left and right hip joints

These metrics were averaged per subsequence and visualized. As shown in Figure 4.3, the two occupant classes formed visually separable clusters in the resulting 2D and 3D spaces.

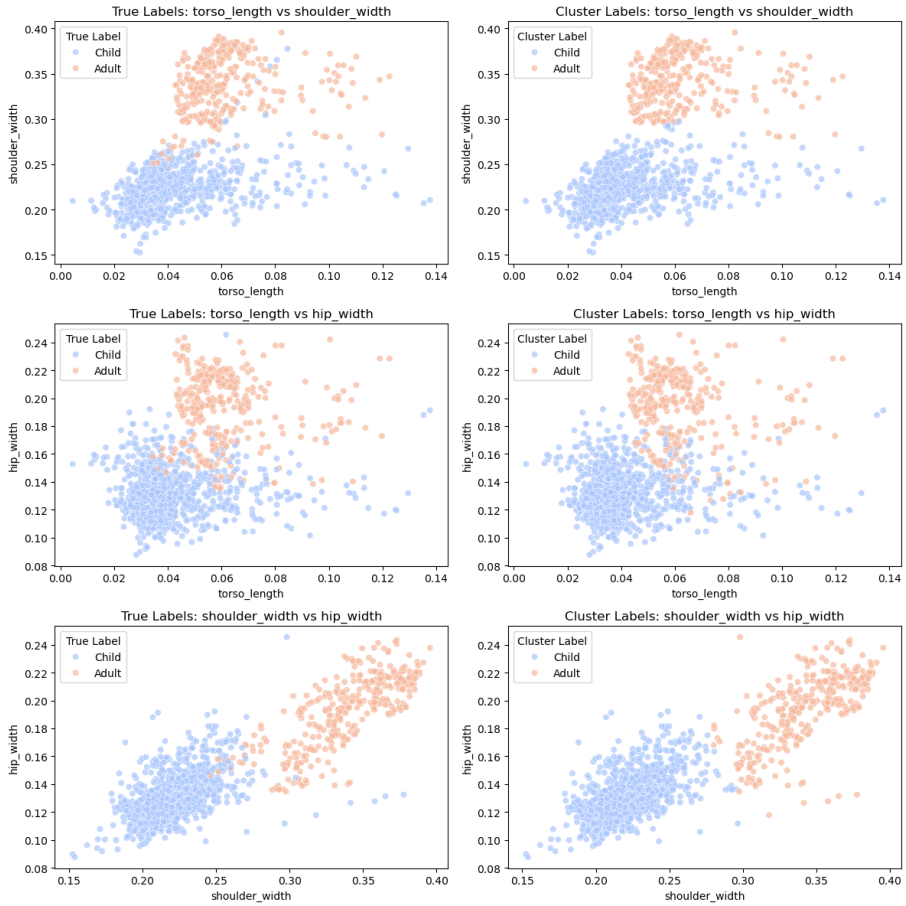


Figure 4.3: Clustering using body metrics: torso length, shoulder width, and hip width. Clusters correspond to Adult and Child-Forward classes.

4.7.2 Clustering on All 3D Keypoints

To analyze clustering in higher-dimensional space, we applied UMAP for dimensionality reduction followed by KMeans clustering ($k = 2$). The choice of $k = 2$

was guided by the silhouette method, which yielded the highest silhouette score of 0.625, indicating optimal separation and cohesion at this cluster count. This also aligns with the known number of ground truth classes, facilitating meaningful evaluation. Pelvis-centered normalization was applied prior to dimensionality reduction. Cluster labels were aligned with ground truth using majority voting. The following unsupervised clustering metrics were recorded:

- **Adjusted Rand Index (ARI):** 0.9789
- **Normalized Mutual Information (NMI):** 0.9520

Figure 4.4 visualizes the UMAP projection colored by true labels, predicted cluster assignments, and clustering errors.

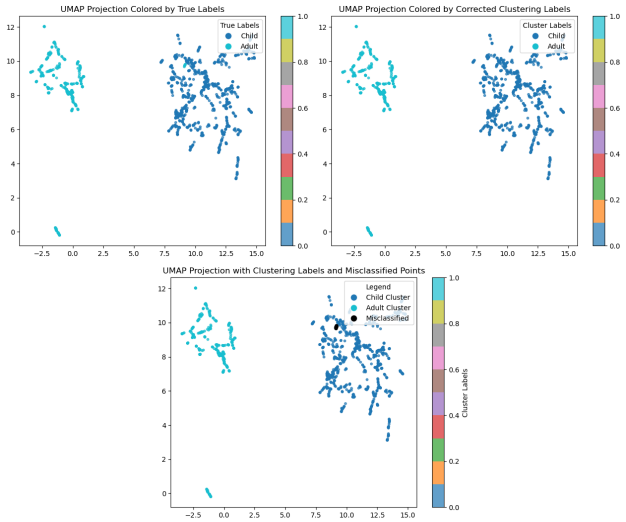


Figure 4.4: UMAP projection of normalized 3D keypoints. Top-left: true class labels; top-right: KMeans assignments; bottom: misclassified points (black).

4.7.3 Error Analysis of Misclassified Sequences

To investigate clustering errors, we inspected misclassified sequences. All errors involved adult samples misclustered as children. These samples exhibited forward-leaning postures with reduced torso height and narrower shoulder width.

Figure 4.5 presents a mean 2D skeleton overlay, comparing misclassified adults (red), true children (blue), and correctly clustered adults (green).

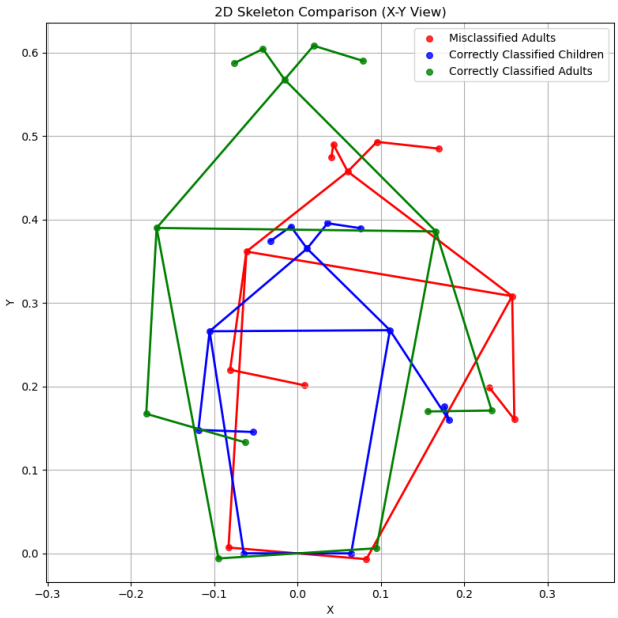


Figure 4.5: Mean 2D Skeleton Comparison (X-Y View). This plot compares average 2D skeleton projections for three groups: misclassified adults (red), correctly classified children (blue), and correctly classified adults (green). Each skeleton is plotted using key anatomical landmarks, where the lower points represent hip joints, the extended lateral points form arms (elbows and wrists), and the upper structure includes shoulders, nose, eyes, and ears. The visual separation in scale and proportions particularly in limb length and shoulder spread helps illustrate the spatial features the model uses and struggles with when classifying body types.

Figure 4.6 quantifies this posture-based ambiguity across three body metrics. Misclassified adults show intermediate geometry between true children and adults.

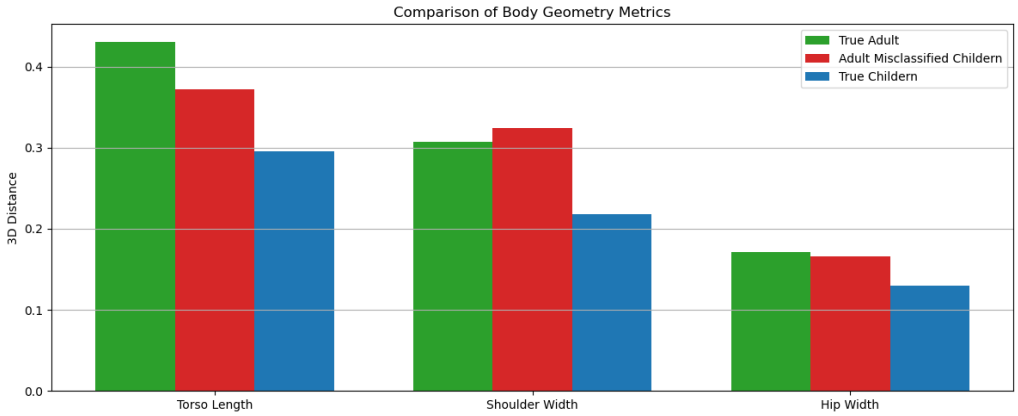


Figure 4.6: Comparison of body geometry metrics across groups. Misclassified adults lie between true children and true adults in torso and shoulder dimensions.

4.8 Regression-Based Sample Size Estimation

The model architecture used in this section is identical to the ResNet-50 + LSTM configuration described in Section 4.3. However, the sample size estimation approach is model-agnostic and can be applied to any classifier supervised or deep including traditional models such as SVM or logistic regression, as long as performance metrics (e.g., F1-scores) are available across different training sizes.

To estimate how much additional training data might be needed to reach higher classification performance, we applied four nonlinear regression models Power Law, Arctangent, Root, and Logarithmic to the observed F1-scores. The data used came from training ratios of 15%, 30%, 50%, 60%, and 75%, and extrapolations were made to predict F1-scores at 87% and 100% training levels.

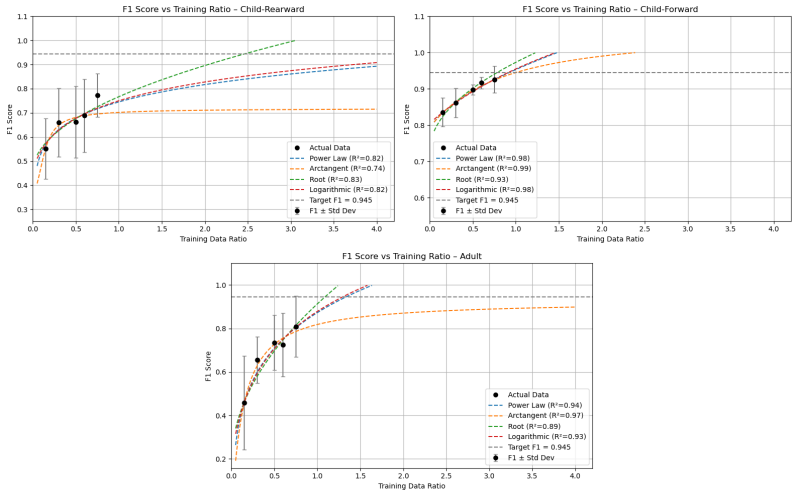


Figure 4.7: Regression-based extrapolation of F1-score as a function of training data ratio for Child-Rearward, Child-Forward, and Adult. The target F1-score in figure is 0.945

These models were fitted per class and excluded F1-scores from 87% and 100% during fitting to simulate an out-of-sample prediction task. The regression was based on performance from the ResNet-50 + LSTM model, as detailed in Section 4.3.

4.8.1 Estimated vs. Actual Additional Frames

To determine how much additional data is required to achieve a target F1 score, we model the F1 score as a function of the data sampling ratio using curve fitting.

Let:

- r_{required} be the estimated sampling ratio needed to reach the target F1 score.
- $r_{\max}$ be the highest sampling ratio used in current training.
- S_{current} be the current average number of training frames for a class.

- S_{needed} be the estimated number of frames needed.

We compute the scale factor:

$$\text{Scale Factor} = \frac{r_{\text{required}}}{r_{\text{max}}}$$

Then the total frames needed are:

$$S_{\text{needed}} = \text{Scale Factor} \times S_{\text{current}}$$

The number of additional frames required is:

$$\Delta S = S_{\text{needed}} - S_{\text{current}}$$

To convert additional frames into the number of additional passengers:

- Let $F_{\text{per passenger}}$ be the median number of usable frames per passenger.
- Then, the number of additional passengers needed is:

$$P_{\text{additional}} = \frac{\Delta S}{F_{\text{per passenger}}}$$

This estimation assumes that data quality and class distribution remain consistent when new passengers are added.

Tables 4.9, 4.10, and 4.11 compare the extrapolated data requirements (as data ratios) with the actual increases observed between training data increments. For each class and target F1-score, the R^2 of the regression fit, the model-estimated required data ratio, the actual ratio increase, and the resulting efficiency ratio are reported. **Efficiency ratios are color-coded: red indicates overestimation, while blue indicates underestimation.**

Table 4.9: Estimated vs. Actual Required Ratios – Child-Rearward

Target F1	Model	R^2	Required Ratio	True Ratio	Efficiency Ratio
0.779 (87%)	Power Law	0.82	1.86	1.23	1.51 (Over)
	Arctangent	0.74	2.67	1.23	2.17 (Over)
	Root	0.83	1.44	1.23	1.17 (Over)
	Logarithmic	0.82	1.73	1.23	1.41 (Over)
0.7964 (100%)	Power Law	0.82	2.19	1.07	2.04 (Over)
	Arctangent	0.74	2.67	1.07	2.48 (Over)
	Root	0.83	1.60	1.07	1.49 (Over)
	Logarithmic	0.82	2.02	1.07	1.88 (Over)

Table 4.10: Estimated vs. Actual Required Ratios – Child-Forward

Target F1	Model	R^2	Required Ratio	True Ratio	Efficiency Ratio
0.9383 (87%)	Power Law	0.98	1.15	1.17	0.98 (Under)
	Arctangent	0.99	1.21	1.17	1.04 (Over)
	Root	0.93	0.98	1.17	0.84 (Under)
	Logarithmic	0.98	1.14	1.17	0.97 (Under)
0.9349 (100%)	Power Law	0.98	1.11	1.13	0.98 (Under)
	Arctangent	0.99	1.16	1.13	1.02 (Over)
	Root	0.93	0.95	1.13	0.84 (Under)
	Logarithmic	0.98	1.10	1.13	0.97 (Under)

Table 4.11: Estimated vs. Actual Required Ratios – Adult

Target F1	Model	R^2	Required Ratio	True Ratio	Efficiency Ratio
0.8240 (87%)	Power Law	0.94	1.09	1.19	0.92 (Under)
	Arctangent	0.97	1.41	1.19	1.19 (Over)
	Root	0.89	1.03	1.19	0.87 (Under)
	Logarithmic	0.93	1.08	1.19	0.91 (Under)
0.8455 (100%)	Power Law	0.94	1.19	1.12	1.06 (Over)
	Arctangent	0.97	1.81	1.12	1.62 (Over)
	Root	0.89	1.10	1.12	0.98 (Under)
	Logarithmic	0.93	1.17	1.12	1.04 (Over)

4.9 Impact of Cluster-Based Risk Assessment on Model Performance

The model architecture used in this section is identical to the ResNet-50 + LSTM configuration described in Section 4.3, ensuring consistent architecture while evaluating the impact of cluster-based risk-aware data augmentation.

4.9.1 Training Strategy Overview

All models were trained for up to 25 epochs with early stopping triggered after 10 consecutive epochs without improvement in macro F1-score. Each configuration was trained three times across five folds, and results were averaged.

We compared the following configurations:

- **Baseline:** Standard model trained without any oversampling.
- **With Review Emphasis:** Medium-risk sequences, identified via cluster-based labeling, were selectively oversampled to enhance generalization. No high-risk (`review`) clusters were found in this experiment.
- **Oversampled-11.4%:** A control experiment where 11.4% of training samples were randomly oversampled, matching the volume increase of the review-emphasis setting.

- **With Review Emphasis (Error Ratio Excluded):** An ablation configuration in which medium-risk clusters were defined without using the validation error ratio criterion.

4.9.2 Cross-Fold Performance Comparison

Figure 4.8 shows macro F1-scores across five folds for each training configuration, alongside average scores with standard deviation. The model trained with medium-risk oversampling (*with_review_emphasis*) achieved higher F1-scores in multiple folds.

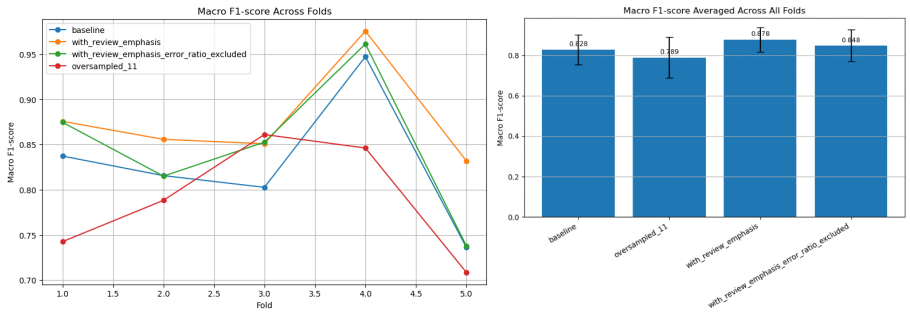


Figure 4.8: Macro F1-score across five folds. Risk-based oversampling (*with_review_emphasis*) outperforms both the baseline and random oversampling. Removing the error ratio criterion reduces performance.

4.9.3 Misclassification Pattern Analysis

Each validation misclassification was categorized based on its relation to the most similar training sample:

- **All Different**: Predicted label differs from both the true label and that of the most similar training sample.
- **Similar = Predicted**: The model prediction matches the label of the most similar training sample, but both are incorrect.
- **Similar = True**: The similar training sample shares the correct label, yet the model misclassifies the input.

Figure 4.9 presents the distribution of these categories across folds.

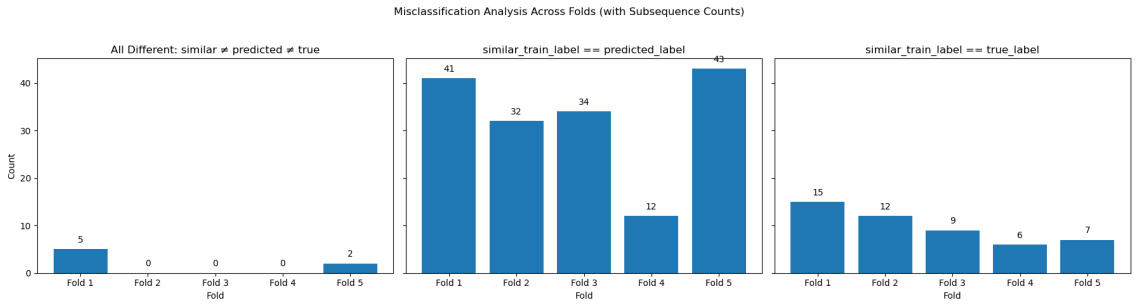


Figure 4.9: Misclassification breakdown across folds. Most errors occur where the predicted label matches a similar (but incorrect) training sample.

5

Discussion

This chapter discusses the findings presented in the previous section, examining the performance and limitations of the models and methods used. It begins with reflections on the evaluation of the baseline model and continues by analyzing the effectiveness of radar heatmap classification using CNN–LSTM architectures. The discussion further addresses the integration of multimodal data, performance of supervised models, and insights gained from 3D keypoint-based classification and clustering. Additionally, the utility of regression-based sample size estimation and risk-aware strategies is evaluated.

5.1 Motivation from Deployed Model Evaluation

The evaluation of the production model currently deployed in the vehicle (Section 4.1) revealed significant performance limitations, particularly in detecting backward-facing children (Class 4), where the model achieved a recall of just 1% and an F1-score near zero. Even for other occupant classes, such as forward-facing children and adults, the model exhibited low recall and precision, resulting in an overall accuracy of only 44%. These shortcomings motivated a deeper investigation into model design, training strategies, and data requirements.

This poor performance can largely be attributed to limitations in the training data. Specifically, the production model was trained on data collected over a relatively short period (approximately 1.5 weeks), which resulted in an imbalanced dataset with **insufficient representation of certain occupant classes** most notably, backward-facing children. This lack of diversity in the training data significantly impaired the model's ability to generalize across all occupant types.

These findings served as the foundation for the development of our custom CNN–LSTM architectures and the introduction of key experiments, such as extrapolation-based sample size estimation and cluster-based risk-aware oversampling. The aim was twofold: (1) to understand the limits of performance imposed by the original data and modeling approach, and (2) to explore practical methods to improve classification, particularly for the most error-prone occupant classes.

5.2 Radar Heatmap Classification with CNN–LSTM

Across all three architectures ResNet-18 + LSTM, ResNet-50 + LSTM, and TinyCNN + LSTM a consistent trend was observed: increasing training size led to clear improvements in both macro-averaged F1-score and accuracy. This highlights the importance of diverse training sets for generalization. Models trained with only 15% of the data exhibited greater variability across folds and lower accuracy, underlining their sensitivity to distribution shifts.

ResNet-18 showed steady improvements, particularly stabilizing around the 50% mark. The Adult and Child-Forward classes achieved stronger performance throughout, while the Child-Rearward class remained more variable despite overall gains. ResNet-50 benefited more significantly from increased data, consistently outperforming ResNet-18 across all training sizes. Its deeper architecture enabled it to reach peak performance earlier often exceeding ResNet-18's best results with just 50% of the data. Notably, some ResNet-50 folds at 75% outperformed those at 100%, suggesting that fold composition and class balance can be as influential as raw data volume.

TinyCNN + LSTM, despite its minimal size, scaled surprisingly well. It achieved competitive results, especially for the more challenging Child-Rearward class, making it suitable for edge-device deployment. However, its performance was more variable at lower data sizes, exposing the limitations of compact models when learning complex temporal patterns from sparse data.

These results emphasize the trade-off between model complexity and training data availability. While deeper networks benefit from more data, smaller models remain viable in constrained settings. Moreover, the importance of well-structured cross-validation cannot be overstated, as fold variability significantly affects outcomes when data is limited.

5.3 Multimodal CNN–LSTM (Heatmaps + Keypoints)

Both multimodal CNN–LSTM architectures ResNet-18 + LSTM with full 3D keypoints and with body metrics demonstrated that fusing radar heatmaps with pose-based features improves classification performance across all training sizes. The inclusion of joint-level pose information or coarse body geometry led to higher macro F1-scores and accuracy compared to radar-only models, particularly for classes **Adult** and **Child-Rearward**.

At 100% training, the radar + 3D keypoints model achieved the highest overall accuracy (0.9080) and macro F1-score (0.8794), outperforming all radar-only baselines including ResNet-50 + LSTM (0.8919 / 0.8670); see Table 4.5 and Table 4.3. This result highlights the value of multimodal fusion even when abundant radar data is available.

However, the **Child-Rearward** class does not benefit from multimodal fusion, as no

reliable keypoint information is available due to occlusion from rear-facing seats. As a result, improvements driven by keypoint features are primarily limited to the **Adult** and **Child-Forward** classes, where the fusion of modalities enhances separability and robustness.

While the full keypoint model introduces higher preprocessing and inference overhead, it offers the strongest performance. The body metrics variant is more lightweight and showed comparable or superior performance in mid-sized training regimes (e.g., 50%–75%), making it a practical alternative when computational efficiency is a priority.

In summary, multimodal fusion consistently outperforms unimodal radar models across data regimes except for classes where keypoint data is unavailable. Full 3D keypoints offer the best overall performance, while body metrics strike a balance between efficiency and accuracy. The choice of modality should therefore consider deployment constraints, class visibility, and data availability.

5.4 Supervised Learning on Flattened Radar Heatmaps

Traditional supervised classifiers trained on flattened radar heatmaps continued to show strong and scalable performance as training data increased. Macro F1-scores steadily improved from 15% to 100% training size, with decreasing variance across folds indicating increased stability with larger datasets.

Among the models, **SVM** consistently delivered the best overall performance, achieving a macro F1-score of **0.8629** at full training size; see Table 4.6. Logistic Regression and XGBoost also performed competitively, especially at lower training sizes, suggesting that the radar-derived features exhibit a degree of linear separability. In contrast, Random Forest and k-NN generally lagged behind likely due to their sensitivity to high-dimensional inputs and lack of spatial reasoning.

Compared to the best-performing deep model, **ResNet-50 + LSTM**, SVM achieved superior per-class F1-scores for both the **Child-Forward** and **Adult** classes, reaching **0.9601** and **0.9092**, respectively, at full training size; see Table 4.7. These scores are on par with those of the multimodal model using 3D keypoints, which achieved **0.952** and **0.893** on the same classes; see Table 4.4. ResNet-50 + LSTM on Radar/multimodal, however, outperformed SVM on the more challenging **Child-Rearward** class, achieving **0.836, 0.792** versus **0.7195**. Despite this, ResNet-50 + LSTM ultimately attained a slightly higher overall macro F1-score (**0.867** vs. **0.8629**), benefiting from its ability to leverage spatiotemporal features.

Class-wise, **Child-Forward** was consistently the easiest to classify across all models, with high performance even at smaller training sizes. **Adult** classification showed steady improvement with more data, particularly under SVM and multimodal models. **Child-Rearward**, while consistently the most difficult class, exhibited its strongest gains when modeled with deep temporal architectures.

In summary, traditional supervised classifiers remain effective and practical baselines, especially in settings constrained by computation or latency. Nevertheless, deep

sequence and multimodal models offer better generalization particularly for complex or underrepresented classes when sufficient training data is available.

5.5 3D Keypoint-Based Subsequence Classification

Experiments on sequence-averaged 3D keypoints demonstrated that classical models can reliably distinguish forward-facing children from adults without temporal modeling. Logistic Regression and CatBoost achieved the highest test-set performance, validating that even linear models are effective when spatial pose features are well-normalized and centered.

Vertical features from the pelvis, ears, and shoulders emerged as the most discriminative, reflecting structural and postural differences between children and adults. These patterns confirm that simple anatomical cues carry strong class-separating signals.

A key limitation was the exclusion of rear-facing children due to occlusion-related keypoint errors. This points to the broader challenge of pose-based approaches under visibility constraints, motivating the need for fusion or occlusion-aware alternatives in future work.

5.6 Clustering Analysis of 3D Keypoints

Unsupervised clustering confirmed that body proportions torso length, shoulder width, and hip width can effectively separate Child-Forward from Adult occupants, even without labels. UMAP + KMeans yielded high agreement with ground truth (ARI = 0.9789, NMI = 0.9520), validating the presence of structure in pose-derived representations.

Importantly, clustering uncovered edge cases. Misclassified adults typically displayed forward-leaning postures that distorted their proportions, making them visually similar to children. Mean skeleton overlays and metric comparisons showed that these individuals occupy a middle ground between class distributions, suggesting inherent ambiguity.

This analysis not only complements supervised learning but offers tools for interpretability, anomaly detection, and targeted data improvement strategies.

5.7 Regression-Based Sample Size Estimation

Nonlinear regression models were applied to estimate the training data required to achieve specific F1 targets. Despite high R^2 values in fitting, extrapolation reliability varied.

In the Child-Rearward class, most models overestimated data needs, with Root

regression proving more conservative and accurate. For Child-Forward, estimates were close to actual increases, especially for Arctangent and Logarithmic models. The Adult class showed mixed results, with Arctangent producing large overestimates.

These findings illustrate that:

- High R^2 does not guarantee extrapolation accuracy, especially near saturation.
- Efficiency ratio is a more robust metric to assess extrapolation quality.
- Conservative models like Root or Logarithmic may be better suited for performance forecasting in high-F1 regions.

Importantly, performance trends were not strictly monotonic, revealing the impact of fold variability or label noise. These challenges highlight the need for extrapolation approaches that account for uncertainty.

5.8 Risk-Aware Oversampling via Clustering

Risk-based oversampling, guided by clustering and validation misclassifications, significantly improved model performance over both baseline and randomly augmented counterparts. The inclusion of a validation-driven error ratio in the risk labeling was especially impactful.

The ablation experiment showed that removing the error-based condition degraded performance, confirming the utility of integrating both training difficulty and error linkage in augmentation strategies. Random oversampling, despite matching in volume, failed to replicate the benefit emphasizing the value of targeted sample selection.

Misclassification analysis further clarified model weaknesses. Most errors arose when the model aligned with similar but incorrectly labeled training samples, indicating representational confusion. Sequences with correct analogs that were still misclassified present prime candidates for retraining. “All different” cases were rare and likely correspond to outliers.

These insights support the use of risk-aware augmentation as a cost-effective enhancement to supervised pipelines, with potential for integration into active learning and semi-supervised training frameworks.

6

Conclusion

This work explored a comprehensive range of radar- and vision-based occupant classification strategies under varying data regimes, model complexities, and fusion strategies. Across all experiments, increasing training data consistently improved model performance, especially for underrepresented or ambiguous classes such as Child-Rearward.

Radar-based deep learning models, particularly ResNet-50 + LSTM, demonstrated strong generalization when enough data was available. Meanwhile, smaller architectures like TinyCNN + LSTM offered competitive performance in resource-constrained scenarios. Multimodal fusion of radar heatmaps with 3D keypoints or body metrics consistently improved classification performance across all training sizes. The radar + 3D keypoints model achieved the highest accuracy and macro F1 even at full training size, surpassing all radar-only baselines.

Classical ML models trained on either flattened radar heatmaps or averaged 3D keypoints proved surprisingly effective, despite their low complexity. For radar-based classification, SVM performed particularly well especially on the Adult and Child-Forward classes achieving strong per-class F1-scores and demonstrating robust generalization when sufficient training data was available. Given its efficiency and minimal configuration requirements, SVM is a compelling choice for radar-based deployment in resource-constrained settings.

In the keypoint-based pipeline, the Child-Rearward class was excluded due to occlusion: rear-facing children are not reliably detected by the camera because their bodies are obstructed by the seat. Despite this limitation, models trained on averaged 3D keypoints still achieved very high accuracy in distinguishing between the remaining classes, confirming the discriminative power of geometric features. Clustering analyses further supported this by revealing meaningful posture-driven variation and identifying edge cases.

Additional contributions include a regression-based framework for estimating data requirements and a cluster-based risk-aware augmentation strategy, both of which offer practical tools for efficient model development and deployment planning.

Taken together, these findings emphasize the importance of model selection based on deployment constraints, data availability, and target class distributions. Future work should explore multimodal sensor fusion under occlusion, domain adaptation across vehicle types, and real-time uncertainty modeling for deployment-ready systems.

Bibliography

[1] R. J. Glass and J. D. Graham, "Kids at risk: Where american children sit in passenger vehicles," *Journal of Safety Research*, vol. 30, no. 1, pp. 17–24, 1999.

[2] M. Skolnik, *Introduction to Radar Systems* (Electrical engineering series). McGraw-Hill, 2001, ISBN: 9780071181891. [Online]. Available: https://books.google.se/books?id=Y6-APwAACAAJ.

[3] M. Roberts, *A programmer's introduction to processing imaging radar data*, [Online; accessed 28-Mar-2025], 2023. [Online]. Available: https://www.plextek.com/articles/a-programmers-introduction-to-processing-imaging-radar-data/.

[4] H. Lichtinger, B. M. Curtis, R. Graf, D. Reich, S. Morrell, and M. Kremer, *Sensor assembly for seat occupant weight classification system*, US Patent 7,503,417, Mar. 2009.

[5] U.S. Department of Transportation, *49 cfr § 571.208 - occupant crash protection*, https://www.ecfr.gov/current/title-49/subtitle-B/chapter-V/part-571/subpart-B/section-571.208, Accessed May 22, 2025, 2025.

[6] M. Vamsi and K. Soman, "In-vehicle occupancy detection and classification using machine learning," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1–6. DOI: 10.1109/ICCCNT49239.2020.9225661.

[7] S. D. Da Cruz, B. Taetz, O. Wasenmüller, T. Stifter, and D. Stricker, "Autoencoder based inter-vehicle generalization for in-cabin occupant classification," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 1296–1303. DOI: 10.1109/IV48863.2021.9575641.

[8] J. P. Van Marter, A. G. Dabak, A. V. Mani, S. Rao, and M. Torlak, "A deep learning approach for in-vehicle multi-occupant detection and classification using mmwave radar," *IEEE Sensors Journal*, vol. 24, no. 20, pp. 33 736–33 750, 2024. DOI: 10.1109/JSEN.2024.3450432.

[9] H. Abedi, M. Ma, J. He, J. Yu, A. Ansariyan, and G. Shaker, "Deep learning-based in-cabin monitoring and vehicle safety system using a 4-d imaging radar sensor," *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11 296–11 307, 2023. DOI: 10.1109/JSEN.2023.3270043.

[10] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, p. 160, 2021, ISSN: 2661-8907. DOI: 10.1007/s42979-021-00592-x. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x.

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[12] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient knn classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018. DOI: 10.1109/TNNLS.2017.2673241.

[13] A. Mammone, M. Turchi, and N. Cristianini, "Support vector machines," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 3, pp. 283–289, 2009.

[14] S. Ji and Y. Xie, "Logistic regression: From binary to multi-class," *Texas A&M University: College Station, TX, USA*, 2024.

[15] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, 1999. [Online]. Available: https://api.semanticscholar.org/CorpusID:12744045.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, ISSN: 1476-4687. DOI: 10.1038/nature14539. [Online]. Available: https://doi.org/10.1038/nature14539.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[19] D. Ogaga and A. Olalere, *Evaluation and comparison of svm, deep learning, and naïve bayes performances for natural language processing text classification task*, Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Connectivity-of-the-Pooling-and-Kernels-in-CNN_fig2_375869038 [accessed 26 Mar 2025], Nov. 2023. DOI: 10.20944/preprints202311.1462.v1.

[20] Y. Zhang, L. Peng, G. Ma, M. Man, and S. Liu, "Dynamic gesture recognition model based on millimeter-wave radar with resnet-18 and lstm," *Frontiers in Neurorobotics*, vol. 16, p. 903 197, 2022, eCollection 2022, ISSN: 1662-5218. DOI: 10.3389/fnbot.2022.903197. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2022.903197/full.

[21] I. E. Livieris, E. Pintelas, and P. Pintelas, "A cnn–lstm model for gold price time-series forecasting," *Neural Computing and Applications*, vol. 32, no. 23, pp. 17 351–17 360, 2020, ISSN: 1433-3058. DOI: 10.1007/s00521-020-04867-x. [Online]. Available: https://doi.org/10.1007/s00521-020-04867-x.

[22] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] S. Mascarenhas and P. Agarwal, "Comparative analysis of cnn architectures for image classification tasks," *International Journal of Engineering and Technology*, 2021.

[25] Evidently AI Team. "How to interpret a confusion matrix for a machine learning model." (Jan. 9, 2025), [Online]. Available: https://www.evidentlyai.com/classification-metrics/confusion-matrix (visited on 04/29/2025).

[26] Evidently AI. "Accuracy, precision, and recall in multi-class classification." (Jan. 9, 2025), [Online]. Available: https://www.evidentlyai.com/classification-metrics/multi-class-metrics (visited on 04/29/2025).

[27] R. Mahmood, J. Lucas, D. Acuna, *et al.*, "How much more data do i need? estimating requirements for downstream tasks," *arXiv preprint arXiv:2207.01725*, 2022.

A

Appendix 1