13

A 3D Simulation Framework for Behavior Cloning on Edge AI-Enabled E-Scooters in Smart Cities

Artemis Stefanidou[1], Eleni Tsaousi[1], Elena Politi[1], Ihsan Can Yalabuk[2], Emrecan Bati[3], Burak Tüfekçi[2], and George Dimitrakopoulos[1]

[1]Harokopio University of Athens, Greece
[2]TÜBİTAK B İLGEM, Türkiye
[3]HOP, Türkiye

Abstract

Modern mobility systems are rapidly advancing through automation, connectivity, and remote operation technologies. Across land and sea, autonomous and teleoperated vehicles are redefining safe, efficient, and sustainable transportation. This work presents a 3D simulation framework for training and evaluating Behavior Cloning (BC) models on edge-AI-enabled micromobility platforms. The framework integrates physics-based simulation with lightweight visual-policy learning to enable autonomous navigation under limited computational resources. The best configuration achieved a weighted F1 score of 0.99 and maintained near- real-time performance on CPU-only setups (44.7 ms per frame, 22 FPS, 1.55 GB RAM), demonstrating the practicality of efficient learning-based control for next-generation e-scooters and smart urban mobility systems.

Keywords: behavior cloning, e-scooter, autonomous vehicles, path planning.

13.1 Introduction and Background

Modern mobility systems are undergoing a significant transformation through the integration of automation, connectivity, and remote operation technologies [1]. Across land, sea, and air, autonomous vessels, aerial drones, teleoperated cars, and intelligent e-scooters are reshaping the way people and goods are transported in urban and maritime domains. These emerging solutions aim to improve safety, efficiency, and sustainability while enabling new modes of transportation that reduce the need for human control onboard [2]. This technological shift has been further accelerated by advances in edge computing, low-latency communication (e.g., 5G/6G), and lightweight Artificial Intelligence (AI) models, which now allow intelligent agents to operate with real-time perception and control even on resource-constrained embedded platforms [3]. Edge intelligence reduces reliance on cloud infrastructure, increases resilience to communication delays, and supports autonomous operation in bandwidth-limited environments.

However, many existing mobility architectures still rely on rich multisensor setups, such as RGB-D cameras, LiDAR, IMU, and radar, and on complex perception–planning pipelines originally designed for larger autonomous vehicles. Even recent Foundation Models (FMs), despite their advances in unified and semantically rich perception, remain computationally demanding and unsuitable for low-power embedded platforms [4]. As a result, such high-capacity configurations become impractical for compact micro-mobility systems, including e-scooters and small delivery robots, which operate under strict constraints in weight, cost, power consumption, and onboard computational capacity [5]. These challenges motivate the exploration of simplified perception modalities and task-specific visual encodings that enable efficient, real-time decision-making without heavy sensor fusion or GPU-intensive processing.

Moreover, the same principles extend beyond ground and maritime mobility. Aerial and underwater vehicles increasingly adopt similar hybrid architectures that combine autonomy with teleoperation, especially in complex or safety-critical environments. Drones, in particular, share the same constraints on onboard computation, energy efficiency, and perception under uncertain conditions. As a result, lightweight image-based and imitation-driven frameworks offer a unified methodology for intelligent navigation across land, air, and sea domains.

Despite this progress, there remains a gap in exploring minimalist sensory modalities in particular, using extremely simplified visual representations

(e.g., binary, black-and-white encoding) to drive navigation policies. However, transferring visually trained navigation policies from simulation to real micro-mobility platforms remains a major challenge due to the well-known sim-to-real gap. Recent studies show that discrepancies in sensing fidelity, motion blur, and actuation nonlinearities can significantly degrade real-world performance [6]. Similar domain gaps have also been documented in autonomous-driving perception pipelines, where models trained on CARLA synthetic data struggle to generalize to real-world lane geometry and appearance without explicit domain-adaptation mechanisms [7]. If such methods prove viable, they could dramatically lower the barrier to deploying autonomous or teleoperated systems on micro-mobility platforms (such as e-scooters), where payload, power, and compute budgets are extremely limited.

In this context, the primary objective of this work is to develop a Behavior Cloning (BC) model that can serve as a pretrained policy for subsequent Reinforcement Learning (RL) stages. Such a model enables the agent to start from an informed behavioral prior rather than learning entirely from scratch, accelerating convergence and improving stability in complex environments.

Building on these innovations, our work is organized around four key contributions:

- **Image-Based Autonomous Pipeline with Binary Path Encoding:** Proposal of a novel learning pipeline for autonomous or teleoperated navigation using binary image input.
- **BC from Human Demonstrations:** Development and training of a BC model that learns navigation policies directly from human driving trajectories, enabling data-efficient imitation without requiring reward signals or environment-specific tuning.
- **Lightweight Binary Image Backbone for Onboard Inference:** Integration of a compact CNN-based (or transformer-based) visual backbone that processes binary (black-and-white) image input, extracting minimal yet informative spatial features optimized for real-time inference on resource-constrained micro-mobility platforms.
- **Path Planning through End-to-End Imitation:** Direct mapping from binary image input to control commands for safe and efficient navigation, bypassing classical feature extraction.

The complete implementation, including training scripts, simulation setup, and BC models, is available as open-source code at: https://github.com/icsa-hua/Ecomobility---Behavior_Cloning_Model.git.

The rest of the paper is structured as follows: Section II reviews related work on BC and imitation learning for autonomous navigation. Section III details the proposed framework, including the e-scooter modelling, dataset preparation, and model architectures. Section IV presents the experimental evaluation and discusses the obtained results. Finally, Section V concludes the paper and outlines future research directions.

13.2 Related Work

A variety of studies have examined e-scooters from multiple perspectives, reflecting their growing role in modern micro-mobility systems. Prior work has analysed their contribution to sustainable transportation, highlighting reduced congestion, emissions, and environmental impact [8]. Additional research has explored their integration into urban transport networks, emphasizing infrastructure compatibility and user behavior [9]. On the autonomy side, several prototypes have investigated core capabilities such as stabilization, obstacle detection, GPS-based navigation, and high-precision localization [10]. At the control level, dynamic modelling and safety remain key topics: PD-based controllers have been used to study riderless e-scooter balancing [11], and teleoperation frameworks have been proposed to improve responsiveness and collision avoidance [12].

However, while these efforts demonstrate substantial progress in understanding and automating e-scooter behavior, research that applies learning-based control, and in particular *behavior cloning (BC)*, to micro-mobility platforms remains limited. To situate our work within the broader imitation-learning landscape, we next review the main variants of BC that have emerged in autonomous navigation. As illustrated in Figure 13.1, existing approaches can be broadly grouped into three principal categories: (i) *Foundational and Extended BC*, (ii) *Hybrid and End-to-End BC*, and (iii) *Applied and Human-Centric BC*. The following paragraphs summarize representative methods in each category.

For **Foundational and Extended BC**, early works focused on direct imitation from expert demonstrations, where neural networks learned control policies by mapping sensory observations to actions [13]. While these approaches achieved data efficiency, they often suffered from covariate shift and poor generalization in unseen conditions. To overcome these limitations, advanced variants such as DAgger and Generative Adversarial Imitation Learning (GAIL) introduced expert feedback and adversarial

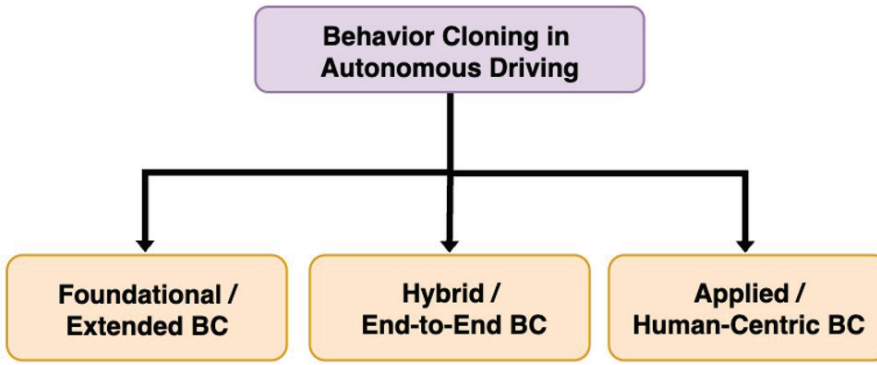

Figure 13.1 Behavior-Cloning-based Categories.

optimization mechanisms to enhance robustness and trajectory alignment [14]. These methods form the theoretical basis for modern imitation learning systems.

Hybrid and End-to-End BC approaches emerged to combine the data efficiency of imitation learning with the adaptability of reinforcement learning, leveraging the strengths of both paradigms. In such frameworks, BC typically serves as a pretraining stage that accelerates policy convergence and stabilizes exploration during reinforcement fine-tuning [15]. Representative examples include DDPG+BC, TD3+BC, and Forward-Prediction-Based Active Exploration BC [16], which integrate exploration mechanisms for improved sample efficiency. In parallel, end-to-end visual learning pipelines emerged, using convolutional or transformer-based backbones to jointly learn perception and control [17], leading to more generalizable and interpretable models suitable for embedded systems [18]. This comparative setup enables quantifying the performance trade-offs between compact CNNs, efficient mobile models, and large-scale foundation transformers in terms of accuracy, latency, and model footprint

Finally, Applied and Human-Centric BC research extends imitation principles to real-world systems such as micro-mobility and teleoperated vehicles. Prototype systems, including e-scooters and mobile robots, have demonstrated that imitation-driven controllers can ensure stable, safe, and adaptive navigation in constrained environments [10], [19]. Similarly, miniature autonomous vehicles have been used as low-cost testbeds to validate real-world BC steering and throttle prediction under resource-constrained settings [20]. Beyond technical control, socially aware models incorporate

Table 13.1 Feature Comparison Across BC Categories and The Proposed Method

Feature	Foundational / Extended BC	Hybrid / End-to-End BC	Applied / Human-Centric BC	Proposed Method
Binary Image Input (B/W Encoding)	✗	✗	✗	✓
Lightweight Edge-Ready Architecture	✗	✓	✗	✓
RL Pretraining Compatibility	✗	✓	✗	✓
Unified Autonomy and Teleoperation	✗	✗	✓	✓
Cross-Domain Adaptability (Land / Air / Sea)	✗	✗	✓	✓
Open-Source and Reproducible Framework	✗	✗	✗	✓

human comfort and perceived safety through mechanisms such as time-to-collision estimation and social force modelling [21-23]. These advances highlight the transition of BC from theoretical imitation learning toward context-aware, deployable.

Overall, the literature reveals an evolution from pure BC toward hybrid, perception-integrated, and socially aware learning architectures. This progression motivates the development of lightweight, binary image-based imitation pipelines capable of real-time navigation on resource-constrained micro-mobility platforms, as pursued in the proposed system. The comparison in Table 13.1 summarizes the distinguishing characteristics of foundational, hybrid, and applied BC frameworks relative to the proposed approach.

13.3 Behavior Cloning Framework Description

The objective of this work is to learn an autonomous navigation policy through BC, enabling a strong initial policy that can later be fine-tuned using RL techniques.

Given a dataset of expert demonstrations

$$\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N} \tag{13.1}$$

where s_i denotes the observed state (e.g., images) and a_i the corresponding control action (e.g., steering, throttle, brake), the goal is to learn a mapping

$$\pi_\theta : s \rightarrow a \tag{13.2}$$

parameterized by θ, which minimizes the discrepancy between predicted and expert actions. The policy network π_θ is trained in a supervised manner by minimizing a loss function:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell\left(a_i, \pi_\theta\left(s_i\right)\right) \tag{13.3}$$

where $\ell(\cdot)$ corresponds to the mean squared error (MSE) for continuous actions or cross-entropy loss for discrete action spaces.

13.3.1 e-Scooter Modeling

The vehicle model adopted in this work represents a compact, lightweight electric scooter designed for autonomous navigation in urban micro-mobility environments. The 3D geometry of the vehicle, along with its coordinate system definition, is illustrated in Figure 13.2.

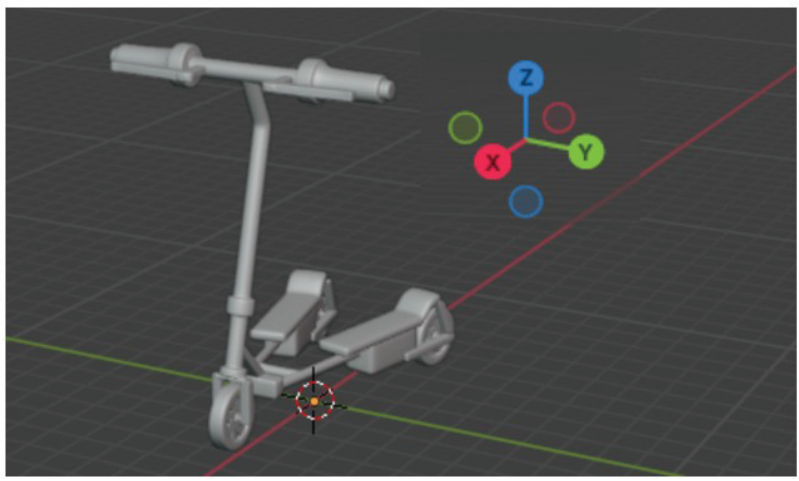

Figure 13.2 e-scooter geometric model and coordinate system definition. The 3D model was created in Blender and integrated into the CARLA simulation environment.

The model was created in Blender, allowing precise control of physical proportions, wheelbase, and was subsequently exported into the simulation environment for dynamic and perception-based experiments.

Unlike conventional two-wheeled scooters, the adopted configuration includes two rear wheels that ensure static stability even when the vehicle is stationary. As a result, the system maintains lateral equilibrium without requiring active balance control. The vertical and lateral forces acting on the scooter are therefore assumed to be in static equilibrium $\sum F_y = 0$, and roll dynamics are neglected. The vehicle is thus modeled as a planar rigid body operating on the horizontal plane, where only longitudinal motion and yaw rotation are considered.

To ensure physical realism and safe operation, the control limits, including maximum acceleration, braking deceleration, and steering angle, are defined according to the mechanical constraints of commercial e-scooters. Moreover, the vehicle's velocity is constrained to comply with the speed limits defined in the CARLA Town01 environment, as shown in Figure 13.3, preventing unrealistic motion or collisions due to excessive speed. This formulation preserves physical plausibility while maintaining computational efficiency, enabling stable and reliable policy learning within the CARLA simulation platform.

Figure 13.3 Speed-limit sign in the CARLA Town01 environment defining the maximum allowed vehicle velocity.

13.3.2 Framework Overview

The proposed BC framework follows a modular design composed of four main stages:

1) **Data Collection and Preprocessing:** Extraction of state–action pairs from simulation or expert teleoperation.
2) **Feature Representation:** Encoding of multimodal sensory inputs using pretrained backbones such as ResNet, MobileNet, or DINOv2 to obtain compact visual embeddings.
3) **Policy Learning:** Training of a lightweight policy head (e.g., MLP or XGBoost) on top of the embeddings to predict expert actions, either as a classification or regression problem depending on the action space.
4) **Evaluation:** Assessment on unseen driving scenarios using success rate, trajectory deviation, and collision count as performance metrics.

This BC setup serves as a foundation for subsequent RL fine-tuning, where the pretrained policy π_θ^{BC} provides a warm start, improving convergence speed and stability during online interaction with the CARLA environment.

Figure 13.4 illustrates the overall BC architecture employed for autonomous navigation. The input to the network corresponds to a 2D projection of the desired path, providing a compact and interpretable representation of the vehicle's intended trajectory. This projection is processed by a convolutional neural network (CNN) backbone, such as ResNet or MobileNet, which extracts high-level spatial features encoding curvature, direction, and local geometric structure. The resulting feature vector is passed to a fully connected classification head that predicts one of four discrete control commands:

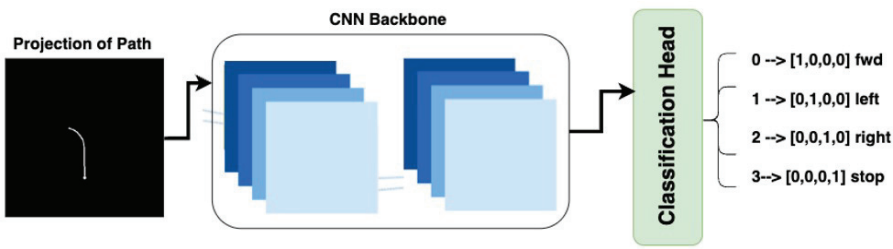

Figure 13.4 Behavior-Cloning-based learning pipeline for autonomous navigation in CARLA

forward, left, right, or *stop*. Each control command is represented as a one-hot encoded vector, serving as the target label during supervised training. This classification-based formulation simplifies the control space and enables efficient learning of navigational behaviors directly from visual inputs.

To investigate the trade-off between representational power, computational efficiency, and generalization capability, two complementary model configurations were developed within this framework.

a) *Deep Policy Network (PolicyNet).* This configuration employs a pretrained visual backbone followed by a lightweight multi-layer perceptron (MLP) head. The backbone is interchangeable, supporting architectures such as ResNet50, MobileNetV3, and DINOv2-ViT-B/14. These networks act as high-level feature extractors, encoding semantic and geometric cues from the input trajectory images, while the MLP head maps the resulting embeddings to discrete control actions. This end-to-end differentiable design enables joint policy optimization with minimal architectural complexity, ensuring efficient deployment on embedded hardware (e.g., NVIDIA Jetson).
   The selection of visual backbones was driven by the goal of covering a broad spectrum of feature representations, from conventional convolutional encoders to recent foundation vision transformers, thus enabling a systematic comparison between compact task-specific models and large-scale pretrained architectures. ResNet50 serves as a well-established CNN baseline, offering strong spatial feature extraction and proven robustness in visual navigation tasks. MobileNetV3 was selected as a lightweight, computationally efficient architecture suitable for real-time inference on resource-constrained platforms, balancing accuracy and latency. In contrast, DINOv2-ViT-B/14 represents a modern foundation model trained on massive self-supervised datasets, capturing semantically rich and highly transferable visual representations. By incorporating both CNN-based and transformer-based backbones, this configuration enables a comprehensive evaluation of whether large-scale pretraining enhances policy transferability and generalization to unseen driving scenarios within the CARLA environment.
b) *Hybrid Embedding + Gradient Boosting Model (ResNet50 + XGBoost).* In this configuration, visual embeddings are first extracted from a frozen ResNet50 backbone and subsequently used to train an XGBoost classifier. This hybrid approach combines the strong representational capacity of deep convolutional features with the interpretability and robustness

of tree-based ensembles. By decoupling feature extraction from policy learning, it allows faster training and flexible experimentation with tabular learning algorithms even on CPU-only setups.

Overall, the two complementary architectures, covering traditional CNNs, efficient mobile models, and foundation vision transformers, enable a unified and systematic comparison, helping to identify which configuration offers the best trade-off between robustness, efficiency, and generalization in behavior-cloned driving policies.

13.3.3 Dataset Preparation

The dataset was collected within the Town01 map of the CARLA simulator, using the in-built GlobalRoutePlanner and LocalRoutePlanner modules to

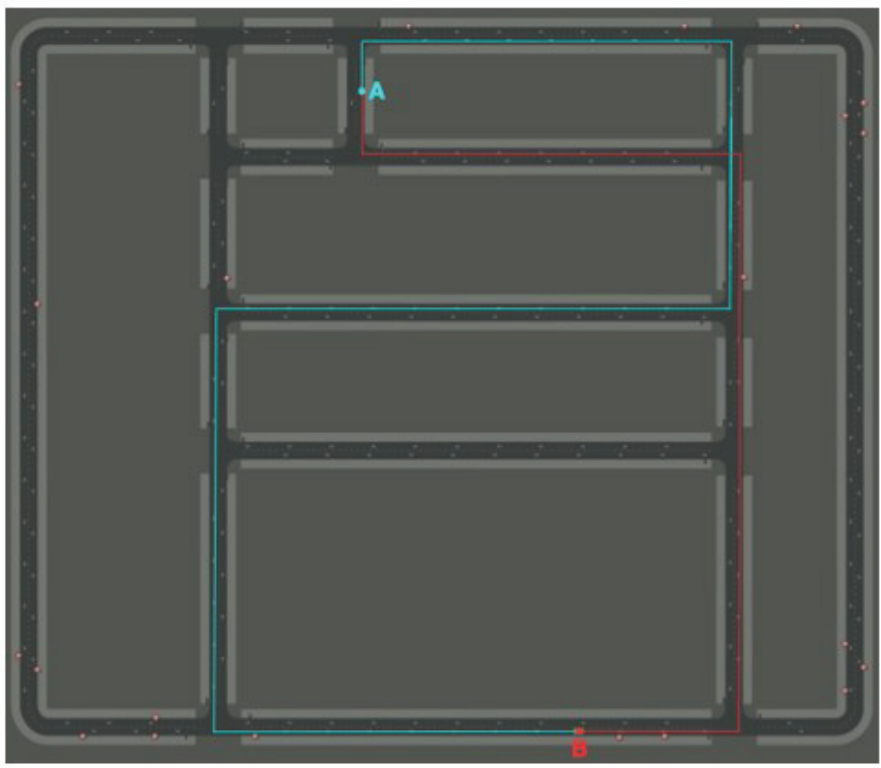

Figure 13.5 Bidirectional routes (A→B and B→A) in CARLA Town01, used to ensure balanced left and right turns in the collected dataset.

generate structured navigation trajectories. A set of ten waypoints ahead of the vehicle was visualized to provide local context, allowing the model to anticipate upcoming turns and adjust its velocity accordingly.

Each sample in the dataset consists of:

- a grayscale image representing the predicted path projection, where white pixels indicate the intended trajectory to be followed by the model;
- a one-hot encoded action label corresponding to one of four discrete control commands: *forward*, *left*, *right*, or *stop*;
- and the file path to the associated image stored in a structured CSV file.

Two routes were recorded within the CARLA environment, one from point A to point B and another in the reverse direction (B to A), as illustrated in Figure 13.5. This bidirectional setup ensured a balanced distribution of left and right turns, enabling the model to generalize equally well across both maneuver types.

Furthermore, the routes include intersections with traffic lights and stop signs, allowing the vehicle to experience and learn stopping behaviors associated with traffic control elements, as shown in Figure 13.6.

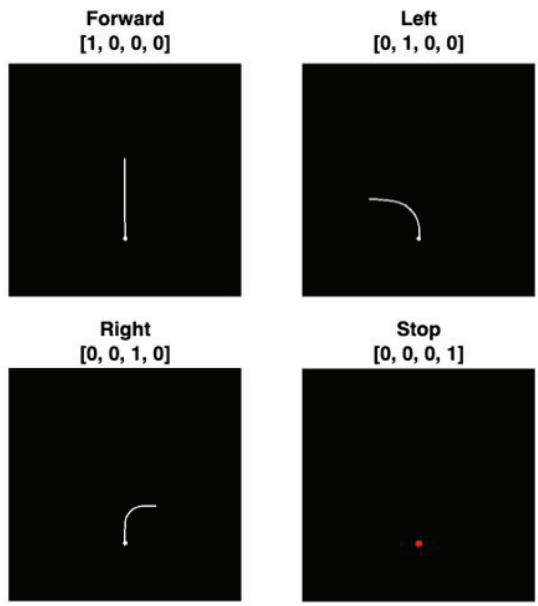

Figure 13.6 Four discrete control commands (forward, left, right, stop) with their one-hot encoded labels. White paths indicate the intended trajectory; the red dot denotes a stop position.

The resulting dataset therefore captures a diverse range of driving contexts, combining geometric cues, directional variation, and traffic-related stopping events, providing a well-rounded foundation for supervised BC.

13.3.4 Evaluation Metrics

Model performance was assessed using standard multi-class classification metrics, complemented by computational profiling to evaluate runtime efficiency.

Performance Metrics: For a classification task with C classes, let TP_c, FP_c, and FN_c denote the true positives, false positives, and false negatives for class c. Each metric is defined as follows:

- **Accuracy:** proportion of correctly predicted actions among all samples.

$$Accuracy = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} (TP_c + FP_c + FN_c)} \tag{13.4}$$

- **Precision:** correctness of positive predictions for class c.

$$Precision_c = \frac{TP_c}{TP_c + FP_c} \tag{13.5}$$

- **Recall:** fraction of correctly identified samples for class c.

$$\mathrm{Recall_c} = \frac{TP_c}{TP_c + FN_c} \tag{13.6}$$

- **F1-score:** harmonic mean of precision and recall.

$$\mathrm{F1_c} = 2\frac{\mathrm{Precision_c} \times \mathrm{Recall_c}}{\mathrm{Precision_c} + \mathrm{Recall_c}} \tag{13.7}$$

- **Macro and Weighted Averages:** global scores across classes.

$$\mathrm{Macro - F1} = \frac{1}{C}\sum_{c=1}^{C} \mathrm{F1_c} \tag{1.8}$$

$$\mathrm{Weighted - F1} = \frac{1}{N}\sum_{c=1}^{C} n_c \times \mathrm{F1_c} \ ,$$

where n_c is the number of samples in class c and

$$N = \sum_{c=1}^{C} n_c \tag{23.9}$$

The confusion matrix $\mathbf{M} \in \mathbb{R}^{C \times C}$ summarizes prediction consistency:

$$M_{ij} = number\ of \text{ samples with true label } i \text{ and predicted as } j$$

Computational Profiling: In addition to accuracy-based metrics, the following efficiency indicators were logged:

- **Memory Usage:** peak CPU/GPU memory consumption.
- **Model Size:** storage footprint of the trained policy.
- **Inference Speed:** average processing time per frame and corresponding frame rate

$$\text{FPS} = \frac{1}{Average\ Inference\ Time} \tag{23.10}$$

Together, these metrics provide a balanced evaluation of predictive accuracy and computational efficiency, reflecting the model's suitability for real-time autonomous navigation.

13.4 Experimental Evaluation

All experiments were conducted on a workstation equipped with the hardware specifications shown in Table 13.2. The dataset was divided into **75%** for training, **15%** for validation, and **10%** for testing, ensuring stratified sampling across all four control actions (*forward, left, right, stop*).

Tables 13.3 and 13.4 summarize the training and test performance of all evaluated models. Among them, **ResNet50+MLP** achieved the best overall accuracy (99.6%) and weighted F1-score (0.996), followed closely by **ResNet50+XGBoost** and **DINOv2+MLP**. In contrast, **MobileNetV3+MLP** exhibited limited generalization, highlighting the trade-off between efficiency and representational power in lightweight CNNs.

During training, the **ResNet50+XGBoost** configuration reached perfect performance (100% accuracy and F1-score of 1.0), indicating that the hybrid

Table 13.2 System Specifications Used for Model Training and Evaluation

Component	Specification
CPU	Intel Core i7-12700KF (12 cores / 20 threads, 3.6–5.0 GHz)
RAM	32 GB DDR4
GPU	NVIDIA GeForce RTX 3060 (12 GB GDDR6 VRAM)
CUDA Version	12.9
Python Version	3.11

Table 13.3 Comparison of Model Performance on the CARLA Training Set

Model	Accuracy (%)	Macro F1	Weighted F1
ResNet50 + MLP	99.5	0.9910	0.9947
MobileNetV3 + MLP	60.1	0.470	0.624
DINOv2 + MLP	99.1	0.9869	0.9915
ResNet50 + XGBoost	100.0	1.000	1.000

Table 13.4 Comparison of Model Performance on the CARLA Test Set

Model	Accuracy (%)	Macro F1	Weighted F1
ResNet50 + MLP	99.6	0.9936	0.9960
MobileNetV3 + MLP	64.9	0.525	0.659
DINOv2 + MLP	98.4	0.9750	0.9842
ResNet50 + XGBoost	99.2	0.9875	0.9922

model fully captured the patterns of the training data. However, when evaluated on unseen test samples, **ResNet50+MLP** achieved slightly higher accuracy (99.6% vs. 99.2%), suggesting stronger generalization ability. This implies that while XGBoost tends to overfit the training distribution, learning even small, noise variations, the MLP head produces smoother decision boundaries, enabling better transferability to new environments and lighting conditions.

The **DINOv2+MLP** model also demonstrated competitive performance (98.4%), confirming that pretrained foundation models can effectively learn visual cues even from simplified grayscale trajectory maps. Nevertheless, its slightly reduced accuracy compared to ResNet50-based variants can be attributed to the high-level semantic abstraction inherent in foundation models. Since DINOv2 was pretrained on diverse multimodal data, its internal representations prioritize semantic structure over fine-grained geometric detail, which limits its precision when processing low-texture or monochromatic inputs such as synthetic path images.

In contrast, the **MobileNetV3+MLP** configuration exhibited evident underfitting, reaching only 60.1% training accuracy and 64.9% test accuracy. The lightweight architecture of MobileNetV3 provides excellent computational efficiency but lacks sufficient representational capacity to capture subtle differences between visually similar directional cues (e.g., left vs. right turns). As a result, the model fails to converge to a consistent decision boundary, confirming the trade-off between efficiency and discriminative power in compact CNN backbones.

The confusion matrices in Figure 13.7 provide deeper insight into the per-class behavior of each model. For **ResNet50+MLP** and

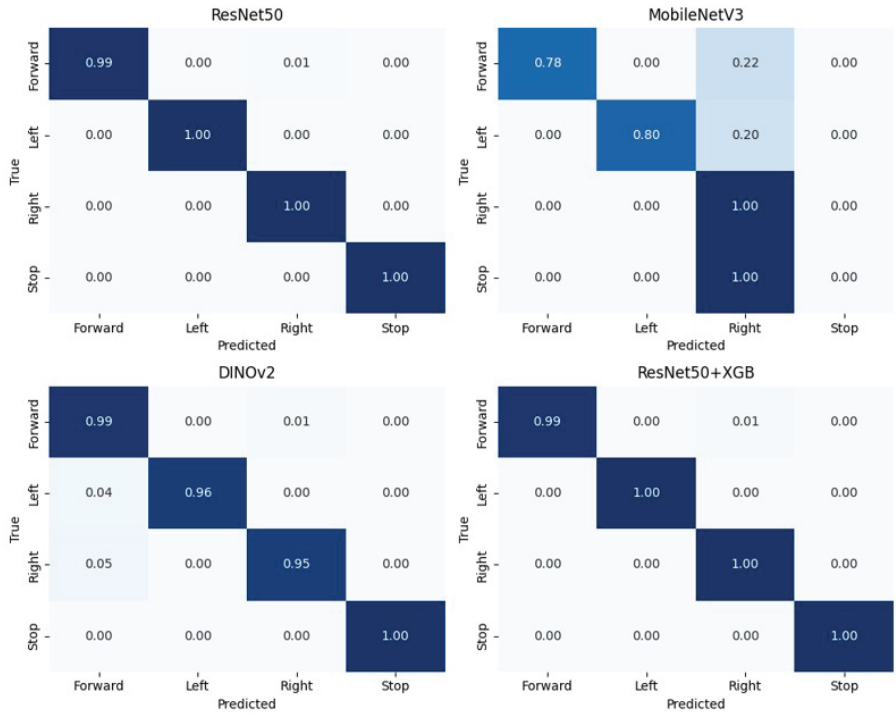

Figure 13.7 Normalized confusion matrices for all evaluated models, including three MLP-based policies (ResNet50, MobileNetV3, DINOv2) and one hybrid approach (ResNet50+XGBoost). Each heatmap shows per-class prediction accuracy for the four control actions (forward, left, right, stop), with strong diagonal values indicating consistent and reliable policy behavior.

ResNet50+XGBoost, all four actions are predicted with near-perfect accuracy, with values above 0.99 across the diagonal, indicating that both models consistently distinguish between straight motion, left/right turns, and stopping. **DINOv2+MLP** also performs robustly, though minor off-diagonal activations reveal that the model occasionally confuses *Left* and *Right* actions (0.04 and 0.05 misclassification rates, respectively). This suggests that while the transformer backbone effectively captures global context, it may overlook small geometric asymmetries in trajectory curvature that are more easily captured by convolutional filters.

In contrast, **MobileNetV3+MLP** shows significant confusion between *Forward* and *Right* commands (22% of forward samples predicted as right). A similar pattern appears between Left and Right, and even the *Stop* class

is entirely misclassified as Right. This indicates that the compact feature maps of MobileNetV3 lack sufficient spatial resolution to encode angular and motion-related cues, resulting in a strong directional bias toward the "Right" action.

Overall, these per-class results validate that higher-capacity models preserve finer geometric distinctions, while smaller backbones trade discriminative strength for efficiency.

After confirming the training consistency and generalization behavior, the three best-performing models were further compared in terms of inference speed on the same GPU-enabled system described in Table 13.2, as shown in Figure 13.8. Among them, the **ResNet50+MLP** configuration achieved the highest throughput (140 FPS), benefiting from its fully end-to-end GPU execution. The hybrid **ResNet50+XGBoost** model achieved slightly lower speed (97.6 FPS) due to the additional CPU-bound inference stage required for the boosting classifier. The **DINOv2+MLP** model was the slowest (89.1 FPS), reflecting the higher computational complexity of its transformer-based backbone.

Since the **ResNet50+MLP** configuration achieved the best overall performance, it was also tested under CPU-only execution to assess its feasibility

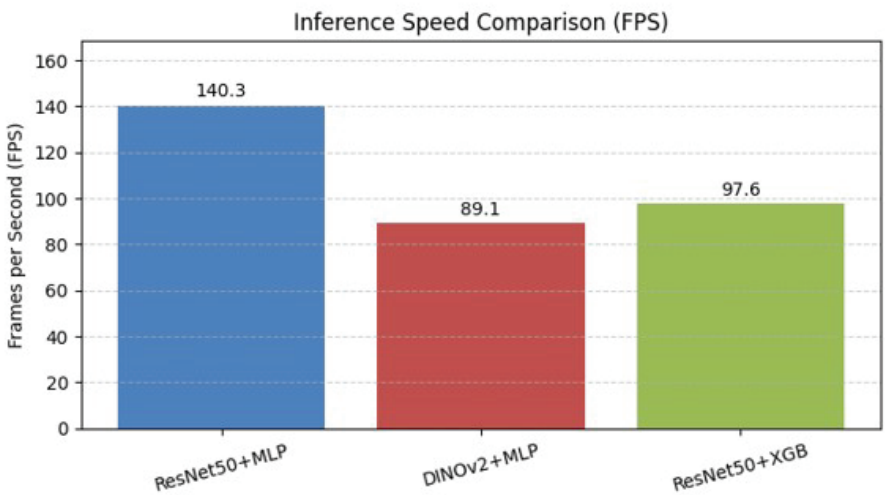

Figure 13.8 Inference speed comparison among the three best-performing models (*ResNet50+MLP, DINOv2+MLP,* and *ResNet50+XGBoost*). The ResNet50-based policy achieves the highest throughput (140 FPS), while all three maintain real-time performance suitable for deployment on embedded platforms.

on systems with limited computational resources. Although benchmarked primarily on a GPU-enabled workstation, the model remains computationally lightweight, requiring only 94 MB of storage, 1.1 GB of RAM, and 288 MB of GPU memory. When executed on CPU-only configurations, the model achieved an average inference time of 44.7 ms per frame (22 FPS) while utilizing approximately 1.55 GB of system memory, confirming its ability to maintain near-real-time performance even on computers with constrained processing capabilities. Such efficiency highlights the potential for deployment not only in e-scooter platforms but also in other low-power autonomous systems, including drones, ground robots, and marine vessels operating under similar hardware constraints.

13.5 Conclusion and Future Directions

This paper presented a comprehensive 3D simulation and learning framework for BC applied to the image-based navigation of edge AI-enabled e-scooters operating in realistic urban environments. The framework integrates a lightweight visual perception pipeline with efficient policy learning mechanisms within the CARLA simulator, allowing autonomous decision-making to be achieved in real time under stringent computational, latency, and energy constraints. The overall design emphasizes scalability, interpretability, and deployability, bridging high-level policy learning with the physics-based simulation of mobility systems. Through this combination, the proposed setup provides a complete digital twin environment for data generation, model training, and safe validation before real-world deployment.

The conducted experiments revealed that compact convolutional and transformer-based visual backbones can effectively learn navigation behaviors from minimal trajectory encodings, essentially binary visual maps, without the need for handcrafted features or complex sensor fusion. The evaluation demonstrated that such compact representations are sufficient to encode critical motion cues, lane geometry, and obstacle boundaries, enabling policies that generalize well to unseen environments while maintaining low inference latency. Among the tested configurations, the **ResNet50+MLP** model achieved the most favourable trade-off between accuracy, inference throughput, and memory consumption. Specifically, it reached 99.6% test accuracy and 140 FPS while requiring less than 1.2 GB of system memory, validating its suitability for real-time control loops in constrained embedded setups. When executed under CPU-only conditions, the model maintained

near-real-time inference performance at 22 FPS and consumed only 1.55 GB of RAM, thereby demonstrating its practicality for deployment on standard computing platforms without dedicated GPUs. This balance of efficiency and predictive performance underlines the potential of such lightweight policies to support energy-efficient autonomy in cost-sensitive or power-limited settings.

Beyond the e-scooter scenario, the proposed pipeline is inherently platform-agnostic and can be seamlessly adapted to other robotic or vehicular domains. The same architecture could be deployed on small aerial drones, autonomous surface vessels, or ground robots, where energy constraints and limited onboard processing power necessitate compact yet robust visual policies. In these contexts, the trained policy network could act as a perception-to-action bridge, transforming raw visual inputs into safe and interpretable motion decisions. Furthermore, its modular nature allows additional perception modalities, such as RGB cameras, depth sensors, LiDAR, or IMU units, to be incorporated into the same learning loop, supporting multi-sensor fusion and richer scene understanding for complex environments.

Such flexibility promotes the development of a unified learning stack capable of operating across land, air, and sea vehicles within the same algorithmic framework.

From a methodological standpoint, the pretrained policies derived through BC can also serve as strong initialization priors for downstream reinforcement learning or safe imitation learning pipelines. This transferability accelerates adaptation to dynamic and uncertain conditions, reducing training time and minimizing safety risks associated with exploration in real-world deployments. In the RL stage, we plan to extend the input representation beyond binary projections by incorporating both grayscale and RGB imagery, allowing the agent to capture richer environmental cues while still benefiting from the lightweight pretraining structure. Future extensions will focus on integrating hybrid Safe Reinforcement Learning schemes that combine risk-aware policy optimization with BC, enabling continuous online learning while preserving safety guarantees. Additional research directions include dynamic obstacle management, cooperative multi-agent path planning, and communication-aware control strategies for interconnected mobility ecosystems.

Overall, this work establishes a scalable, interpretable, and energy-efficient foundation for trustworthy Edge AI autonomy. By unifying visual perception, policy learning, and control within a single 3D digital twin

framework, it contributes to the long-term vision of adaptive, safe, and sustainable intelligent transportation systems that can operate seamlessly across land, air, and sea domains.

Acknowledgements

This work has received funding from the Chips Joint Undertaking (JU) under the Horizon Europe Framework Programme and National Authorities under grant agreement No. 101112306. It has also been implemented under the National Recovery and Resilience Plan "Greece 2.0", funded by the European Union – NextGenerationEU (project name: CoDrones).

References

[1] C. Kettwich, A. Schrank, and M. Oehl, "Teleoperation of highly automated vehicles in public transport: User-centered design of a human-machine interface for remote-operation and its expert usability evaluation," Multimodal Technologies and Interaction, vol. 5, no. 5, 2021. [Online]. https://www.mdpi.com/2414-4088/5/5/26

[2] O. Amador, M. Aramrattana, and A. Vinel, "A survey on remote operation of road vehicles," IEEE Access, vol. 10, pp. 130 135–130 154, 2022.

[3] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," Mobile networks and applications, vol. 26, no. 3, pp. 1145–1168, 2021.

[4] A. Stefanidou, E. Politi, and G. Dimitrakopoulos, "Foundation models in autonomous driving: A review of current tasks and applications," IEEE Open Journal of Intelligent Transportation Systems, vol. 6, pp. 1522–1538, 2025.

[5] A. Musa, H. Kakudi, M. Hassan, M. Hamada, U. Umar, and M. Salisu, "Lightweight deep learning models for edge devices—a survey," International Journal of Computer Information Systems and Industrial Management Applications, vol. 17, p. 18, 01 2025.

[6] M. Yang, H. Cao, L. Zhao, C. Zhang, and Y. Chen, "Robotic sim-to-real transfer for long-horizon pick-and-place tasks in the robotic sim2real competition," arXiv preprint arXiv:2503.11012, 2025.

[7] C. Hu, S. Hudson, M. Ethier, M. Al-Sharman, D. Rayside, and W. Melek, "Sim-to-real domain adaptation for lane detection and

classification in autonomous driving," in 2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022, pp. 457–463.

[8] R. Strässer, M. Seidel, F. Brändle, D. Meister, R. Soloperto, D. H. Ferrer, and F. Allgöwer, "Collision avoidance safety filter for an autonomous e-scooter using ultrasonic sensors," IFAC-PapersOnLine, vol. 58, no. 10, pp.22–28, 2024.

[9] C. Latinopoulos, A. Patrier, and A. Sivakumar, "Planning for e-scooter use in metropolitan cities: A case study for paris," Transportation Research Part D: Transport and Environment, vol. 100, p. 103037, 2021. [Online]. https://www.sciencedirect.com/science/article/pii/S1361920921003357

[10] R. Soloperto, P. Wenzelburger, D. Meister, D. Scheuble, V. S. Breidohr, and F. Allgöwer, "A control framework for autonomous e-scooters," IFAC-PapersOnLine, vol. 54, no. 2, pp. 252–258, 2021.

[11] Y.-H. Lin, A. Jafari, and Y.-C. Liu, "Dynamic modeling and stability analysis of balancing in riderless electric scooters," in 2024 American Control Conference (ACC), 2024, pp. 421–426.

[12] S. Saparia, A. Schimpe, and L. Ferranti, "Active safety system for semi-autonomous teleoperated vehicles," 2021. [Online]. https://arxiv.org/abs/2106.14554

[13] Y. Gao, Y. Liu, Q. Zhang, Y. Wang, D. Zhao, D. Ding, Z. Pang, and Y. Zhang, "Comparison of control methods based on imitation learning for autonomous driving," in 2019 tenth international conference on intelligent control and information processing (ICICIP). IEEE, 2019, pp. 274–281.

[14] G. C. K. Couto and E. A. Antonelo, "Generative adversarial imitation learning for end-to-end autonomous driving on urban environments," in 2021 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2021, pp. 1–7.

[15] Y. Tian, X. Cao, K. Huang, C. Fei, Z. Zheng, and X. Ji, "Learning to drive like human beings: A method based on deep reinforcement learning," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6357–6367, 2021.

[16] H. Xu and D. Zhao, "Forward-prediction-based active exploration behavior cloning," in 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT). IEEE, 2024, pp. 2112–2117.

[17] Y. Fan, "Autonomous driving in unity simulated environments using end-to-end cnn approach," in 2024 7^{th} International Conference on

Advanced Algorithms and Control Engineering (ICAACE). IEEE, 2024, pp.1335–1338.

[18] H.-G. Kim, J. Myoung, S. Lee, K.-M. Park, and D. Shin, "Design of ai-powered hybrid control algorithm of robot vehicle for enhanced driving performance," IEEE Embedded Systems Letters, vol. 16, no. 2, pp. 94–97, 2023.

[19] K. Liu and R. Mulky, "Enabling autonomous navigation for affordable scooters," Sensors, vol. 18, no. 6, 2018. [Online]. https://www.mdpi.com/1424-8220/18/6/1829

[20] P. Moraes, C. Peters, H. Sodre, W. Moraes, S. Barcelona, J. Deniz, V. Castelli, B. Guterres, and R. Grando, "Behavior cloning for mini autonomous car path following," in 2024 IEEE URUCON. IEEE, 2024, pp. 1–5.

[21] A. Jafari and Y.-C. Liu, "Time-to-collision based social force model for intelligent agents on shared public spaces," International Journal of Social Robotics, vol. 16, no. 9, pp. 1953–1968, 2024.

[22] V. Tang and J. Liu, "Mapping urban obstacles: Improving route accessibility for blind and low-vision pedestrians," in 2024 IEEE MIT Undergraduate Research Technology Conference (URTC). IEEE, 2024, pp. 1–5.

[23] S. Gilroy, D. Mullins, E. Jones, A. Parsi, and M. Glavin, "E-scooter rider detection and classification in dense urban environments," Results in engineering, vol. 16, p. 100677, 2022.