DOTTORATO DI RICERCA IN

AUTOMOTIVE ENGINEERING FOR INTELLIGENT MOBILITY

Ciclo 37

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

RACE SMART, LAST LONGER: DEEP LEARNING APPROACHES FOR LI-ION BATTERY STATE ESTIMATION AND AUTONOMOUS RACING VEHICLES

Presentata da: Michael Bosello

Coordinatore Dottorato

Davide Moro

Supervisore

Giovanni Pau

Co-supervisore

Roberto Girau

Esame finale anno 2025

Alma Mater Studiorum – Università di Bologna

Department of Computer Science and Engineering

PhD in Automotive Engineering for Intelligent Mobility

Race Smart, Last Longer: Deep Learning Approaches for Li-ion Battery State Estimation and Autonomous Racing Vehicles

Author
Michael Bosello

Supervisor
Giovanni Pau

Co-supervisor
Roberto Girau

Final Examination Year: 2025

Abstract

As the global demand for advanced autonomous systems continues to grow, reliable energy storage and management becomes paramount, especially in high-performance contexts such as autonomous racing vehicles—including cars and drones. This dissertation explores two interrelated topics: (1) the accurate estimation and prediction of Lithium-ion (Li-ion) battery states, and (2) the development of Autonomous Vehicles (AVs) in competitive racing environments. The integration of these topics underscores the pivotal role that energy management plays in maximizing the operational efficiency, safety, and performance of autonomous systems.

The first subject of the thesis addresses the challenges associated with the state estimation and prediction of Li-ion batteries, which are the cornerstone of energy storage in modern autonomous systems. The accurate estimation of the State Of Charge (SOC), State Of Health (SOH), and Remaining Useful Life (RUL) is critical for ensuring the longevity, reliability, and optimal performance of these batteries, particularly in applications where they are exposed to extreme operational stress. Through the application of handpicked Deep Learning (DL) techniques and the development of novel data-driven models, this research improves the accuracy of battery state estimation and prediction, mitigating the risk of unexpected failures and enhancing the operational lifespan of energy storage systems. These advancements are validated through testing on real-world datasets, demonstrating significant improvements over conventional methods. Moreover, this research has developed a comprehensive dataset of battery data under varied conditions, which has been made publicly available to support further research in battery state estimation.

The second subject of this dissertation focuses on the development and testing of AVs, particularly in the context of high-speed racing environments. Racing provides a unique and challenging testing ground for autonomous systems, where the need for rapid decision-making, precise control, and high performance under dynamic and uncertain conditions is critical. This research contributes to the field by employing deep Reinforcement Learning (RL) techniques for autonomous driving tasks, particularly utilizing LIDAR and 3D-LIDAR sensors for perception. The thesis introduces methodologies to tackle the challenges of sim-to-real transfer, ensuring that models trained in simulation environments can perform in the real world. Additionally, a novel dataset for autonomous drone racing is introduced, which provides a benchmark for aggressive, high-speed navigation tasks.

The third subject of the thesis explores the *potential* for integration between these two domains, which is the unifying theme presented throughout the study. This part highlights the symbiotic relationship between battery performance and vehicle control policies' success. AVs can extend their operational time and improve performances by adjusting control strategies based on real-time battery state information. Conversely, battery health can be preserved and optimized by controlling energy spikes through adaptive driving strategies.

Overall, this dissertation makes contributions to both the fields of battery state estimation and AV control. It presents open-source software and publicly available datasets that support the research community in advancing both these domains. By improving the accuracy of Li-ion battery state estimation and prediction, this research enables more reliable and efficient energy use in autonomous systems. Simultaneously, the advancements in autonomous racing vehicles provide valuable insights into the future of high-performance autonomy, where speed, precision, and reliability are of the utmost importance. These findings lay the groundwork for future research into the convergence of energy management and autonomous system design, promising further innovations in the pursuit of more sustainable and capable autonomous technologies.

List of Contributions

Li-ion Battery State Estimation and Prediction (Sec. 2.1)

Peer-Reviewed Conference Papers

- K. L. Wong, M. Bosello, R. Tse, *et al.*, “Li-ion batteries state-of-charge estimation using deep lstm at various battery specifications and discharge cycles,” in *Proceedings of the Conference on Information Technology for Social Good*, ser. GoodIT ’21, ACM, 2021, pp. 85–90. DOI: 10.1145/3462203.3475878
  Appendix A, Section 2.1.1.

Journal Publications

- M. Bosello, C. Falcomer, C. Rossi, *et al.*, “To charge or to sell? ev pack useful life estimation via lstms, cnns, and autoencoders,” *Energies*, vol. 16, no. 6, 2023. DOI: 10.3390/en16062837
  Appendix B, Section 2.1.2.

Open-Source Software

- Battery SOC Estimation with Deep LSTM
  `github.com/KeiLongW/battery-state-estimation`
  Code implementation of [1] (Appendix A, Section 2.1.1).

- Battery RUL Estimation via LSTM, CNN, and Autoencoder
  `github.com/MichaelBosello/battery-rul-estimation`
  Code implementation of [2] (Appendix B, Section 2.1.2).

Datasets

- UNIBO Powertools Dataset (Battery State Estimation)
  `data.mendeley.com/datasets/n6xg5fzsbv/1`
  Dataset presented and used in [1] (Appendix A, Section 2.1.1).

Autonomous Vehicles in Racing Contexts (Sec. 2.2)

Peer-Reviewed Conference Papers

- M. Bosello, R. Tse, and G. Pau, "Train in austria, race in montecarlo: Generalized rl for cross-track f1tenth lidar-based races," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2022, pp. 290–298. DOI: `10.1109/CCNC49033.2022.9700730`
  Appendix C, Section 2.2.1.

- Y. Chen, R. Tse, M. Bosello, *et al.*, "Enabling deep reinforcement learning autonomous driving by 3D-LiDAR point clouds," in *Fourteenth International Conference on Digital Image Processing (ICDIP 2022)*, vol. 12342, SPIE, 2022. DOI: `10.1117/12.2644369`
  Appendix D, Section 2.2.2.

Journal Publications

- M. Bosello, D. Aguiari, Y. Keuter, *et al.*, "Race against the machine: A fully-annotated, open-design dataset of autonomous and piloted high-speed flight," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3799–3806, 2024. DOI: `10.1109/LRA.2024.3371288`
  Appendix E, Section 2.2.3.

Open-Source Software

- Autonomous Car Racing with F1TENTH and Reinforcement Learning
  `github.com/MichaelBosello/f1tenth-RL`
  Code implementation of [3] (Appendix C, Section 2.2.1).

- Autonomous Car in CARLA Simulator with Reinforcement Learning
  `github.com/MichaelBosello/f1tenth-RL/tree/carla-sim`
  Code implementation of [4] (Appendix D, Section 2.2.2).

- Autonomous Drone Racing Dataset Scripts (Processing, Visualization, Control)
  `github.com/tii-racing/drone-racing-dataset`
  Code implementation of [5] (Appendix E, Section 2.2.3).

Datasets

- RATM Autonomous Drone Racing Dataset
  `github.com/tii-racing/drone-racing-dataset/releases`
  Dataset presented in [5] (Appendix E, Section 2.2.3).

Contents

1 Introduction **1**

1.1 Motivations 3

1.1.1 Li-ion Batteries, the fuel of modern technology 3

1.1.2 Racing as a Benchmark for Autonomy 4

1.1.3 Reciprocal Benefits of Battery Management and Autonomous Control . 5

1.2 Challenges and Related Works 7

1.2.1 Li-ion Battery State Estimation and Prediction 7

1.2.2 Autonomous Vehicles in Racing Contexts 10

1.2.3 Integration of Battery Management and Autonomous Vehicles 12

2 Contributions **15**

2.1 Li-ion Battery State Estimation and Prediction 15

2.1.1 Paper A: Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles 16

2.1.2 Paper B: To Charge or to Sell? EV Pack Useful Life Estimation via LSTMs, CNNs, and Autoencoders 18

2.2 Autonomous Vehicles in Racing Contexts 19

2.2.1 Paper C: Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1^{tenth} LIDAR-Based Races 20

2.2.2 Paper D: Enabling Deep Reinforcement Learning Autonomous Driving by 3D-LiDAR Point Clouds 22

2.2.3 Paper E: Race Against the Machine: a Fully-annotated, Open-design Dataset of Autonomous and Piloted High-speed Flight 24

3 Future Work **27**

3.1 Integration of Battery Management and Autonomous Vehicles 27

3.1.1 Real-Time Energy-Aware Decision Making 27

3.1.2 Predictive Maintenance and Battery Lifespan Optimization 28

3.1.3 Challenges in Sim2Real Transfer and Model Generalization 28

3.1.4 Toward a Unified Framework 28

A Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles 31
A.1 Introduction . . . 33
A.2 Li-ion battery datasets . . . 35
A.2.1 UNIBO Powertools Dataset . . . 35
A.2.2 LG 18650HG2 Li-ion Battery Data . . . 36
A.3 Methodology . . . 36
A.3.1 Recurrent Neural Networks Primer . . . 36
A.3.2 Long Short-Term Memory Primer . . . 37
A.3.3 Proposed LSTM Approach . . . 37
A.3.4 Data Normalization . . . 39
A.3.5 Model Training . . . 39
A.4 Results and discussion . . . 39
A.4.1 UNIBO Powertools Dataset . . . 39
A.4.2 LG 18650HG2 Li-ion Battery Data . . . 41
A.5 Final remarks . . . 43

B To Charge or to Sell? EV Pack Useful Life Estimation via LSTMs, CNNs, and Autoencoders 45
B.1 Introduction and Background . . . 47
B.2 Related Works . . . 50
B.3 Experiments . . . 54
B.3.1 Datasets . . . 54
B.3.2 Models . . . 56
B.4 Results . . . 58
B.4.1 NASA Randomized . . . 58
B.4.2 UNIBO Powertools . . . 61
B.5 Conclusions and Future Works . . . 61

C Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1^{tenth} LIDAR-Based Races 63
C.1 Introduction . . . 65
C.2 Related Works . . . 66
C.2.1 RL for autonomous racing . . . 66
C.2.2 LIDAR-based RL . . . 67
C.3 Reinforcement Learning Primer . . . 68
C.3.1 Modeling the driving problem . . . 69
C.4 System Overview . . . 69
C.4.1 The F1tenth platform . . . 69
C.4.2 RL environment . . . 70
C.4.3 LIDAR data pre-processing . . . 71

C.4.4 From simulation to reality 72
C.4.5 Driver-agent software 72
C.5 Results and Discussion 73
C.5.1 NN comparison experiment 73
C.5.2 Sim2real experiments 75
C.5.3 F1 racetracks experiment 77
C.6 Conclusions and Future works 78

D Enabling Deep Reinforcement Learning Autonomous Driving by 3D-LiDAR Point Clouds 81
D.1 Introduction 83
D.2 Related work 85
D.2.1 Reinforcement Learning in autonomous driving 85
D.2.2 Object detection technology based on LiDAR 86
D.2.3 Related work in CARLA simulator 87
D.3 Methodology 87
D.3.1 Reinforcement Learning 87
D.3.2 PointNet Algorithm 89
D.3.3 Carla Simulation Environment 89
D.4 Experiment 90
D.5 Conclusions 92

E Race Against the Machine: a Fully-annotated, Open-design Dataset of Autonomous and Piloted High-speed Flight 93
E.1 Introduction 95
E.2 Related Work 96
E.3 An Open-design Quadrotor for Autonomous Drone Racing and Data Collection 97
E.3.1 Design Overview 98
E.3.2 Sensors 99
E.3.3 Software 100
E.4 Data Collection Protocol 100
E.4.1 Flight Arena, Racing Gates, and Motion Capture System 100
E.4.2 Flight Program 101
E.4.3 Human-piloted and Autonomous Control 102
E.4.4 Image Labeling 103
E.4.5 Time Synchronization 103
E.4.6 Data Post-processing 103
E.5 Data Format 104
E.5.1 Camera-aligned and Uniform-sampling CSVs 105
E.5.2 Drone and Gates' Poses from Motion Capture 105
E.5.3 Camera Frames 106

E.5.4 Image Labels 106
E.5.5 On-board Data from the Quadrotor Electronics 106
E.5.6 On-board Data from the Autonomous Module 106
E.6 Visualization and Post-processing Scripts 107
E.7 Conclusions 108

Bibliography **111**

Chapter 1

Introduction

The rapid advancement of autonomous systems has driven remarkable innovations in both robotics and energy management, two fields increasingly interconnected as the world embraces more sustainable and efficient technologies. At the core of this progress are Lithium-ion (Li-ion) batteries, which power a wide range of modern autonomous systems, from electric ground vehicles to aerial drones. As those systems rely on Li-ion batteries for energy storage, accurate estimation and prediction of the battery state become critical for ensuring optimal performance, longevity, and safety [6], [7]. The battery State Of Charge (SOC), State Of Health (SOH), and Remaining Useful Life (RUL) emerge as crucial variables for optimizing the productivity and cost of those systems [8]. In parallel, Autonomous Vehicles (AVs)—particularly in high-speed and safety-critical contexts such as urgent delivery, search and rescue missions, and transportation—are pushing the boundaries of Machine Learning (ML) and control algorithms [9], [10]. These systems rely on tightly integrated perception-action loops and real-time decision-making capabilities to navigate complex environments at high speeds. The synergy between energy management and AV control is an emerging field of research poised to transform the performance of autonomous systems in energy-intensive scenarios [11]. The symbiotic relationship between these two interconnected domains is evident when considering how one affects the other. For instance, taking aerial vehicles as an example, it exists an optimal speed at which a quadrotor should operate to minimize energy consumption and extend the operational range [12], resulting in cost reduction during deployment, i.e., by flying faster, drones can be more productive [12]. This is shown by the plot in Fig. 1.1, suggesting the optimal speed is above 15 m/s (54 km/h) for small drones. This dependence is bi-directional as on one hand, the vehicle's control policy affects how efficiently the battery is discharged and how it degrades, and on the other hand, the battery's state affects the vehicle's performance and the control algorithm's effectiveness. In the first case, the vehicle's control policy can be optimized to minimize or limit energy consumption and improve overall system efficiency by adjusting the speed and acceleration profile based on the battery's state. In the second case, the battery's state can inform the vehicle's decision-making process, allowing it to adjust its speed or strategy to conserve energy when the task requires the full use of the battery capacity throughout the mission and maximize performance by balancing battery consumption during the task execution.

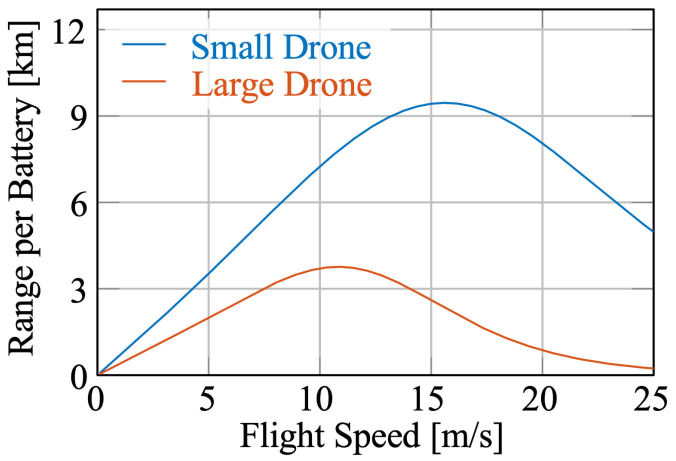


Figure 1.1: Flight range vs. speed for a multicopter, computed by using the full aerodynamics plus the battery model. The optimal speed for energy consumption w.r.t. operational range is above 15 m/s in the case of small drones. Source: [12].

This dissertation explores these two distinct, yet closely related, topics. The first (Sec. 2.1) focuses on the state estimation and prediction of Li-ion batteries using ML, with an emphasis on SOH and RUL. The second (Sec. 2.2) addresses the challenges of autonomous vehicle control in high-performance racing contexts, with contributions to both autonomous car and drone racing, which have been proved to be excellent benchmarks for assessing the state-of-the-art of AVs systems. Both topics are tackled using ML, which has significantly transformed both battery management [13] and autonomous systems [14]. Although no direct work has been done to fully integrate battery management with autonomous control, this dissertation lays the groundwork for future research in this area by enhancing both domains separately, and highlighting the potential for their integration (Sec. 3.1), with the promise to revolutionize the performance and sustainability of autonomous systems. This work introduces novel Deep Learning (DL)-based models for battery state estimation and prediction, Reinforcement Learning (RL)-based control systems for autonomous—racing—cars, and Optimal Control (OC)-based data recording of aggressive autonomous drone flights. It also shares novel publicly available datasets and open-source tools for both battery and autonomy topics. By fostering accurate energy management and sophisticated vehicle control, this thesis provides a pathway for the future development of sustainable and high-performance autonomous systems.

This thesis is organized in the form of a collection of papers. The first part, composed of three introductory chapters, highlights the concepts, motivations, and contributions of this work. It follows the appendix containing the standalone publications. The dissertation evolves as follows: Chapter 1 has introduced the two topics of interest and their union, outlining how the sections will be divided into three parts: (Sec. 2.1) *Li-ion Battery State Estimation and Prediction*; (Sec. 2.2) *Autonomous Vehicles in Racing Contexts*; and (Sec. 3.1) the opportunities for future *Integration of Battery Management and Autonomous Vehicles*. The chapter continues with the motivations behind the research, and an exploration of the challenges and related works, both divided into three parts as well. Chapter 2 presents the contributions of this work, including the development of DL models for battery state estimation (Sec. 2.1), and the advancements to the autonomous racing vehicles field (Sec. 2.2), while Chapter 3 outlines the directions for future work regarding the integration between battery management and autonomous vehicle control (Sec. 3.1), discussing how to move towards accurate battery state estimation that can enhance the performance and reliability of autonomous systems, while also reducing costs and improving sustainability. The appendices include the contributions as self-contained papers, each appendix corresponding to a publication.

1.1 Motivations

The importance of Li-ion batteries in modern technology cannot be overstated. From consumer electronics to Electric Vehicles (EVs) including drones, they provide the energy necessary for the operation of countless devices. Nevertheless, the safety and reliability of batteries are a principal concern, as malfunctions can have severe consequences in both safety-critical and high-performance scenarios [15], [16]. In autonomous systems, the need for accurate and reliable energy management is amplified. Without accurate knowledge of a battery's SOC, SOH, and RUL, autonomous vehicles are prone to unexpected failures.

In parallel, AVs, particularly in racing contexts, represent the cutting edge of robotics and Artificial Intelligence (AI). Racing environments offer a unique opportunity to test the limits of perception, decision-making, and control systems, pushing AVs to operate under extreme conditions of speed and agility [9], [10]. In such environments, every millisecond counts, and the energy efficiency of the system plays a pivotal role in determining its overall performance. The ability to accurately predict the battery's behavior and manage its power output is essential for success in high-speed autonomous tasks.

This thesis addresses these two critical challenges—battery state estimation and high-performance autonomous control—highlighting the interplay between them. By improving battery state estimation, autonomous systems can better manage their energy consumption, enabling more efficient and sustained performance in high-stakes environments such as racing.

1.1.1 Li-ion Batteries, the fuel of modern technology

In the modern technological landscape, Li-ion batteries have emerged as one of the most critical enablers of portable power. Their widespread adoption across a variety of industries—from consumer electronics to EVs and renewable energy storage—is a testament to their superior energy density, longer cycle life, and high efficiency when compared to older battery chemistries such as lead-acid or nickel-cadmium [8]. With the global shift towards electrification and sustainability, the reliance on Li-ion batteries is growing exponentially, making them indispensable in the pursuit of reducing carbon emissions and transitioning away from fossil fuels [17]. Li-ion batteries have revolutionized multiple sectors. In consumer electronics, they power the mobile devices and laptops that have become ubiquitous in daily life. In renewable energy, they are vital for grid storage solutions, helping to balance intermittent sources like solar and wind. However, one of the most transformative impacts has been in the automotive industry, where Li-ion batteries are central to the development of EVs [18], [19]. By providing a lightweight and efficient energy storage solution, they have made EVs a viable alternative to internal combustion engines, supporting global efforts to reduce greenhouse gas emissions.

The operational characteristics of Li-ion batteries, such as their high energy-to-weight ratio and low self-discharge rate, make them well-suited for use in high-performance and energy-critical applications. Nevertheless, their performance is influenced by several factors [6], including temperature, current loads, and the number of charge-discharge cycles, which affect their SOC, SOH, and overall longevity. Accurate estimation of these parameters is crucial, especially in systems like EVs and autonomous robots, where the battery's status directly impacts performance, safety, and reliability. Despite their advantages, managing the complexities of Li-ion batteries remains a significant challenge, particularly in demanding applications [8], [20]. As these batteries age, their capacity to hold charge diminishes, leading to shorter operating times and reduced overall efficiency. This makes accurate SOC and SOH estimation essential for optimizing performance, extending battery life, and preventing failures that could lead to operational downtime or, in extreme cases, hazardous situations such as thermal runaway. In this context, advanced Battery Management Systems (BMS) are integral [6], [7]. These systems leverage sophisticated algorithms, including ML, to predict and manage battery health, ensuring efficient energy use and extending battery life. As the demand for high-efficiency, sustainable energy storage continues to rise, the need for innovation in Li-ion battery technology and management grows more urgent. The research presented in this thesis contributes to this field by developing novel methods for battery state estimation and prediction, aiming to enhance the safety, reliability, and sustainability of systems powered by Li-ion technology.

1.1.2 Racing as a Benchmark for Autonomy

Autonomous racing is not limited to entertainment or competitive sports; it also serves as a critical benchmark for the development of autonomous systems that could have a profound impact on society. Racing environments demand the highest levels of system reliability, speed, and precision, making them an ideal proving ground for autonomous technologies [9], [10]. The challenges faced in autonomous racing, such as real-time decision-making, high-speed navigation, and adaptability in dynamic and unpredictable settings, directly translate into real-world applications where safety and efficiency are paramount. In fact, the development of autonomous systems that can reliably perform under the extreme conditions of racing will likely yield technologies that are even more reliable in everyday applications. For instance, autonomous cars designed for racing are tested for their ability to operate safely at high speeds, manage complex maneuvers, and respond instantaneously to dynamic environments. These capabilities, when transferred to consumer AVs, could enhance safety in normal traffic scenarios [10], where system reliability is crucial for preventing accidents and ensuring smooth operation in various driving conditions. Additionally, advancements in autonomous drone racing, where drones must perform complex tasks with high agility, can lead to the development of systems for industrial inspections, disaster response, and other critical fields that require fast and accurate navigation through challenging environments [9]. To address the broader applicability of autonomous racing technologies, it is important to highlight their potential for generalization to various domains beyond competitive contexts. In urban environments, for example, the real-time perception and decision-making systems developed for racing can be adapted to improve navigation and collision avoidance for self-driving cars, ensuring safe and efficient transport. Similarly, logistics operations could benefit from the precise motion planning and trajectory optimization algorithms honed in racing, allowing autonomous vehicles and drones to navigate warehouses or deliver goods with greater speed and reliability. In search and rescue scenarios, the ability to quickly and accurately maneuver through dynamic environments could significantly improve response times and effectiveness during emergencies. By leveraging advancements in racing, autonomous systems can address complex challenges across a wide range of real-world applications, maximizing their societal impact. Therefore, while autonomous racing may appear to be an isolated niche, it serves as an essential testing ground for innovations that will ultimately benefit society in broader applications. By pushing autonomous systems to their limits in racing contexts, we can ensure that these technologies will perform reliably in everyday scenarios, from transportation and logistics to healthcare and emergency response.

1.1.3 Reciprocal Benefits of Battery Mgmt. and Autonomous Control

The relationship between battery management and autonomous control is reciprocal, with each domain influencing the other in critical ways. Optimizing an AV's control policy to minimize energy consumption and reduce battery strain can result in longer operational durations and extended battery lifespans. Conversely, precise estimation of the battery's state can enhance the vehicle's control decisions, enabling adjustments to speed or strategy to conserve energy, prevent premature battery depletion, and maintain optimal performance throughout the task's execution.

Control Policy Effects on Energy Consumption and Battery Degradation

The control policies governing autonomous systems play a pivotal role in shaping energy efficiency and the rate of battery degradation. In AVs, whether ground-based or aerial, decisions related to speed, acceleration, braking, and maneuvering directly impact energy utilization and, consequently, battery depletion over time. Optimizing these control policies is essential, not only to extend the operational range of the vehicle but also to prolong the overall lifespan of its battery. Control strategies that emphasize aggressive acceleration and high-speed maneuvers may enhance short-term performance but impose a significant load on the battery. High discharge rates lead to increased internal temperatures, accelerating degradation processes such as electrolyte decomposition and electrode wear [6], [21]. Conversely, adopting more conservative control policies—such as limiting acceleration or maintaining steady speeds—can significantly alleviate battery stress. These approaches reduce discharge rates, minimize thermal buildup, and mitigate capacity fade, thereby enhancing the battery's longevity. The control policy further influences battery degradation by the depth and frequency of discharge cycles. Systems operating near their maximum power output often induce deeper discharge cycles, exacerbating wear on battery components. In contrast, smoother driving patterns or flight paths that involve gradual acceleration and deceleration help maintain shallower discharge cycles. This approach ensures balanced energy consumption and slower aging of the battery, extending its functional lifespan [20].

Advanced ML algorithms integrated into the vehicle's control system can further enhance this relationship. By incorporating real-time data on the battery's SOC and SOH, control policies can dynamically adjust to optimize energy use. For example, an AV might reduce its top speed or reroute based on current battery conditions, conserving energy for critical tasks or mitigating rapid battery degradation during demanding operations. This feedback loop between battery state and control policy is essential for achieving both high performance and long-term sustainability. Such adaptive control strategies can be particularly impactful in scenarios that demand rapid decision-making under changing environmental conditions, such as autonomous racing or search and rescue operations. In these high-stakes contexts, where both speed and efficiency are critical, a finely tuned control policy that balances performance demands with energy conservation can provide a competitive advantage by not only preserving battery health but also extending mission capabilities [11].

In conclusion, the control policy of an autonomous system is a key determinant in how efficiently energy is consumed and how quickly a battery degrades. By carefully designing and continuously optimizing these policies, it is possible to significantly enhance both the performance and longevity of battery-powered AVs, thus contributing to more sustainable and efficient system operations.

Battery State Effects on Racing Performance

In high-performance environments such as autonomous racing, the state of the battery plays a pivotal role in determining the vehicle's overall performance. Unlike traditional applications, where energy efficiency is often prioritized over speed, racing environments demand a delicate balance between maximizing power output and conserving enough battery capacity to sustain performance throughout the race. As a result, the battery's SOC, SOH, and RUL have direct implications on the vehicle's ability to compete effectively.

The SOC, which indicates the remaining available energy, is a crucial determinant of how aggressively an autonomous racing vehicle can perform at any given moment. Vehicles with higher SOC can afford to engage in more power-intensive maneuvers such as rapid accelerations, sharp cornering, and overtaking opponents. However, as the SOC depletes, the vehicle must strategically adjust its driving behavior to avoid draining the battery too quickly. Maintaining an optimal SOC throughout the race is essential, as vehicles that fail to manage their energy consumption risk running out of power before the finish line [11] or experiencing a noticeable drop in performance during critical stages of the race.

Equally important is the SOH, which measures the battery's ability to hold charge relative to its original capacity, and affects the vehicle's long-term performance [22]. Batteries with poor SOH exhibit reduced capacity and increased internal resistance, both of which degrade the vehicle's acceleration and top speed. In a racing context, where every millisecond matters, even slight reductions in the battery's ability to deliver power can lead to significant performance penalties. For instance, a vehicle with a degraded battery may struggle to maintain competitive speeds during high-power segments of the race or fail to keep pace with competitors on straightaways. Battery degradation is especially critical in prolonged racing scenarios, where the repetitive cycles of high power output can further erode the SOH. Autonomous racing vehicles that do not account for battery degradation in their control policies may find themselves facing a sudden and sharp decline in performance midway through the race, as the battery's capacity to deliver power diminishes more rapidly than anticipated. Effective energy management, therefore, requires not only optimizing for short-term speed but also considering the long-term health of the battery to ensure consistent performance throughout the event.

In racing environments where vehicles operate at their limits, even small variations in battery state can have profound effects on performance. For instance, the internal resistance of a partially depleted battery can cause voltage drops under high loads, reducing the effective power delivered to the motors. As a result, the vehicle's acceleration and handling may suffer, leading to slower lap times or diminished maneuverability when it matters most. In summary, the battery's state is a critical factor in the overall performance of autonomous racing vehicles. Managing the SOC, SOH, and RUL dynamically during a race allows for strategic adjustments in driving behavior, enabling the vehicle to maintain competitive speeds while preserving sufficient energy for a strong finish. By tightly integrating battery state monitoring with autonomous control policies, it is possible to enhance both performance and battery longevity, creating more resilient and competitive autonomous racing systems.

1.2 Challenges and Related Works

This section provides details on the main challenges of the topics at hand and the limitations of the existing works proposed in the literature, which drive the research presented in this thesis.

1.2.1 Li-ion Battery State Estimation and Prediction

Accurate battery state estimation and prediction are vital for effective energy management in EVs. However, the inherent complexities of battery behavior under real-world conditions pose significant challenges, with existing methods—despite their diversity and continual evolution—exhibiting notable limitations.

SOC Estimation

One of the most explored aspects in the field is the accurate estimation of SOC. SOC represents the available charge within a battery, typically expressed as a percentage of its full capacity, but it cannot be measured directly. Instead, SOC must be inferred indirectly through measurements of external parameters such as voltage, current, and temperature [23]. This estimation is inherently complex due to the highly nonlinear and dynamic nature of the electrochemical reactions occurring within batteries [24]. Inaccurate SOC estimation can lead to operational inefficiencies such as overcharging or over-discharging, both of which significantly shorten battery life and impair EV performance [25]. For our analysis, the methods of SOC estimation can mainly be categorized into direct estimation techniques, battery modeling methods, and ML-based approaches [25]. Direct methods attempt to correlate SOC with physical characteristic parameters, but they often suffer from inaccuracies due to the variability of these parameters across different operating conditions. Battery modeling methods, which rely on mathematical representations of the battery's internal processes, are more sophisticated but require deep knowledge of battery chemistry and physics. These methods, while often accurate under controlled conditions, struggle to generalize to real-world environments due to their reliance on idealized assumptions and complex electrochemical models. ML methods and DL techniques have emerged as a promising alternative due to their ability to model complex, nonlinear relationships without needing a detailed understanding of the internal battery mechanisms [13]. These approaches have been particularly successful in estimating SOC by utilizing datasets of voltage, current, and temperature readings to train neural networks that can predict SOC with high accuracy. For instance, Recurrent Neural Networks (RNNs) and their variant, Long Short-Term Memory (LSTM) networks [26], have shown the capacity to handle sequential data and learn long-term dependencies, making them suitable for SOC estimation [27]. Nevertheless, despite their successes, DL methods face challenges related to the quality and availability of training data, model interpretability, and robustness across different battery types and conditions.

SOH Estimation

SOH estimation, which reflects battery aging and degradation, adds another layer of complexity. Like SOC, SOH cannot be directly measured and must be inferred from indicators such as capacity fade or increases in internal resistance. These indicators are influenced by multiple factors, including temperature, current, and charge/discharge rates [28]. The degradation of Li-ion batteries is a highly nonlinear and multi-faceted process, making it difficult to model accurately. Battery degradation is influenced by both internal factors, such as chemical side reactions, and external factors, such as varying operating temperatures and dynamic charging/discharging profiles [20]. Consequently, electrical and electrochemical models of battery degradation are not only technically demanding but also require extensive experimental data, which may be costly and time-consuming to obtain [29]. A classification of battery SOH estimation methods is detailed in Fig. 1.2.

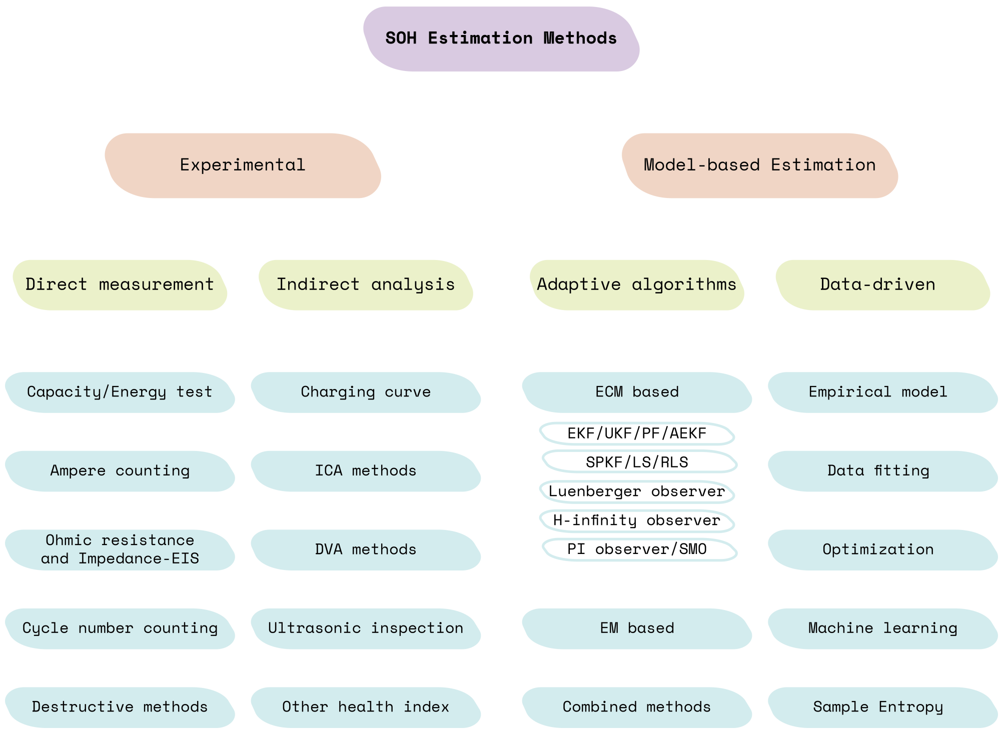


Figure 1.2: Classification of battery SOH estimation methods.

RUL Prediction

RUL prediction, which estimates the remaining useful life of the battery (i.e., SOH forecasting), is a less explored and even more challenging task. An issue that arises when dealing with the RUL lies in the definition of RUL itself. In real-world EV applications, where batteries undergo irregular charging and discharging patterns, the traditional definition of RUL as the number of remaining full cycles before the End Of Life (EOL) [30]–[32] becomes impractical. An alternative metric is proposed in one of the studies presented in this dissertation (Paper B), defining the RUL as the remaining ampere-hour (Ah) capacity before EOL.

SOC/SOH/RUL Methods Limitations

The existing approaches to SOC, SOH, and RUL estimation, despite their improvement in recent years, have several notable limitations.

Physical models, while theoretically sound, often fall short in practical applications. These methods require detailed knowledge of the battery chemistry and rely on assumptions that may not hold under real operating conditions. For instance, many models assume Constant Current (CC) charging and discharging [28], which does not reflect the highly dynamic nature of EV operations, where usage patterns vary according to driving conditions, temperature fluctuations, and user behavior [20], [33]. Besides, those methods typically focus on the current state (e.g. SOC, SOH), with limited capability for long-term prediction (e.g. RUL).

ML-driven approaches, particularly those utilizing DL, offer an alternative by bypassing the need for complex physical models. Neural networks, especially LSTMs, have been widely employed for SOC and SOH estimation due to their ability to capture long-term dependencies in time-series data [13], [28]. LSTMs are particularly effective in handling sequential data, such as voltage and current measurements over time, and have been shown to outperform traditional methods in many cases. However, one of the main limitations of DL-driven methods is their dependence on high-quality training data and comprehensive datasets that reflect real-world driving conditions [28], [33]. Many studies rely on simplified datasets, such as the widely used NASA battery dataset [34], which only includes CC cycling. The use of such datasets limits the generalizability of the trained models to real-world EV operations, where batteries experience dynamic charging and discharging cycles [20]. Another critical shortcoming of DL-based approaches is their limited interpretability. Unlike traditional model-based methods, which are grounded in physical laws and can provide insights into the underlying processes, neural networks operate as “black boxes”, making it difficult to interpret their predictions [35]. This lack of transparency can be concerning in safety-critical applications like EV battery management, where understanding the rationale behind predictions is essential for diagnosing and mitigating potential issues.

Many existing RUL prediction models are flawed by using input data that are not measurable. For example, they rely on historical SOH data which needs to be estimated as well (introducing noise and uncertainty at deployment time), plus they require a warm-up period before they can produce predictions [36]–[38]. This limitation makes them less suitable for real-world applications, where immediate robust predictions are necessary. Furthermore, most RUL models have been developed using oversimplified datasets, e.g., several works focused on a subset of the NASA dataset [38]–[41], which again fails to capture the complexity of real-world battery usage. Traditional RUL models also often consider full charge-discharge cycles from 0% to 100% SOC, which are not representative of typical EV usage.

The variability in battery types and configurations further limits the applicability of existing estimation methods. Many models are tailored to specific battery chemistries, capacities, or conditions. For instance, models trained on small-capacity batteries in laboratory settings often perform poorly when applied to larger-capacity commercial EV batteries. Similarly, models optimized for CC conditions struggle under the dynamic load profiles of real-world operations.

1.2.2 Autonomous Vehicles in Racing Contexts

Autonomous driving and autonomous racing are extensive topics, and a comprehensive review of them is out of the scope of this section. Some excellent surveys exist in the literature, like [10] for autonomous car racing, and [9] for autonomous drone racing (which includes the dataset presented in Paper E). More specialized surveys are also available, like [42] for the application of RL to autonomous driving, and [43] for F1TENTH autonomous racing (which includes Paper C in the selection of RL-based algorithms). This section will focus on the challenges associated with the publications presented in this dissertation, specifically addressing the application of DL and RL in the context of racing and the development of datasets to support their applicability in such a context.

The development of AVs within racing environments presents unique challenges compared to traditional autonomous driving applications. In racing, AVs must operate at high speeds while navigating complex and dynamic environments, often requiring split-second decision-making and precise control. Autonomous racing, whether in small-scale setups like F1TENTH vehicles, larger platforms like full-size cars, or agile robots like drones, introduces complexities across several domains such as perception, planning, and control. Those three elements compose the typical control loop for autonomous robots, as shown in Fig. 1.3. One of the primary challenges is the non-linear dynamics of vehicles at high speeds, which makes the application of ML for control more complicated. Another main challenge is the need for real-time perception and decision-making under conditions of high velocity and limited reaction time. The algorithms must account for real-time processing, requiring high computational efficiency to handle quick decision-making based on sensor inputs like LIDAR or cameras. Additionally, the stochasticity and uncertainty in real-world conditions (e.g., dynamic opponents, track surface changes) introduce further difficulties for the systems. Various approaches have been explored to address these challenges, with DL, RL, and sensor-specific techniques playing critical roles. However, several technical limitations and research gaps remain unresolved in this domain.

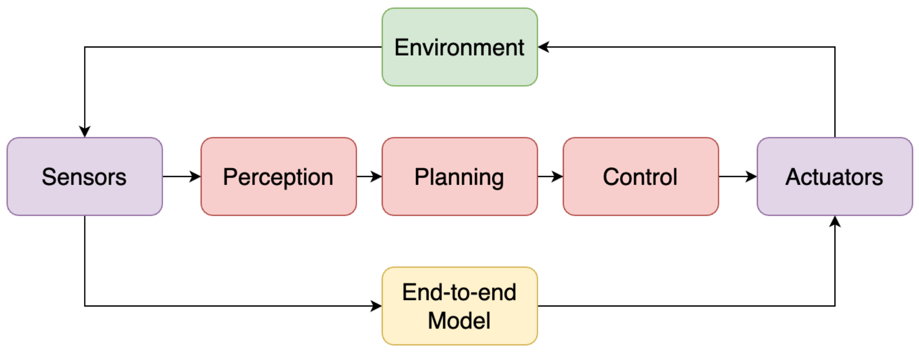


Figure 1.3: Two architectures for autonomous control: (red) the traditional perception-planning-control loop, and (yellow) the end-to-end learning-based control.

Traditional ML models, particularly those leveraging supervised learning, have made significant strides in perception tasks using sensors like LIDAR, RGB cameras, and others. Applications such as lane detection and object recognition are notable examples of their success [44], [45]. However, these models rely heavily on DL approaches that require extensive labeled datasets, introducing substantial computational and infrastructural burdens. Moreover, supervised learning often fails to generalize to dynamic and novel environments, and it often struggles to capture the nuanced decision-making needed for racing. In contrast, RL has emerged as a powerful alternative for autonomous racing [9], [10]. RL enables agents to learn directly through interaction with the environment [46], bypassing the need for labeled data. This trial-and-error learning process makes RL particularly suitable for dynamic and unpredictable settings such as racetracks. RL has shown promise in both end-to-end control and specialized driving tasks like lane-following and overtaking [42]. Despite its advantages, the challenge of generalizing models from virtual simulations to real-world scenarios, commonly referred to as the *sim2real gap*, remains a critical hurdle. Even when models perform well in simulation, they often fail to translate effectively to real-world environments [42] due to differences in data distributions and unforeseen physical complexities, unless specific countermeasures are adopted [14]. Racing contexts further exacerbate this gap, as simulations typically simplify the physical dynamics and interactions that occur at high speeds, which are essential for robust performance. The sim2real gap is further aggravated by the computational constraints of onboard systems, where the RL model must take decisions within milliseconds. While some works have ventured into training RL agents directly in real-world systems [47], [48], safety concerns and the high cost of trial-and-error learning remain barriers to the widespread adoption of such methods.

Aiming to address the challenge of handling aggressive maneuvers and high-speed decision-making, datasets specifically tailored for aggressive drone racing have begun to emerge, providing valuable benchmarks for developing and testing new control algorithms [9], [49]. Those datasets represent a critical factor in the development of autonomous racing systems, as they provide the necessary data to train and evaluate models in realistic racing scenarios. However, many of these datasets still lack the complexity and variability required to fully evaluate the performance of aerial AVs in diverse racing environments. Additionally, existing datasets often rely heavily on simulation, which may not capture the full range of real-world dynamics encountered in physical racing [50].

1.2.3 Integration of Battery Management and Autonomous Vehicles

AVs require not only precise control over their navigation and driving dynamics but also efficient management of energy resources. Batteries are put under significant stress in AVs, as suggested by the plot in Fig. 1.4, which shows how voltage drops during aggressive autonomous aerial maneuvers. Primary challenges in the integration of BMS into AVs revolve around accurate modeling and control of battery degradation, the optimization of energy consumption strategies, and the prediction of vehicle performance under various conditions. The reviewed works explore different aspects of these challenges, providing significant insights into three key aspects.

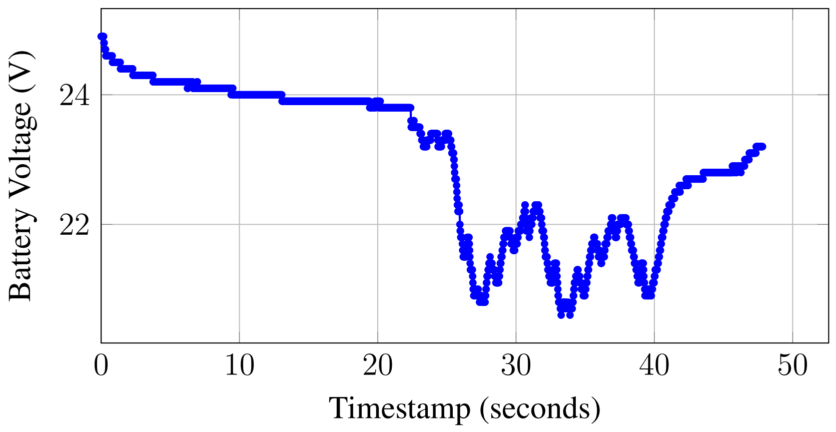


Figure 1.4: Battery voltage readings from a quadrotor, showing voltage drops during aggressive aerial maneuvers. Data recorded in the dataset released in Paper E.

Battery Degradation-Aware Control

Battery degradation presents a significant challenge for AVs, particularly in high-performance contexts like racing, as it significantly affects the performance of autonomous systems. Taking into account the SOC and the RUL of batteries during execution is vital to reduce maintenance costs, avoid safety hazards, and limit operational downtimes. Research has highlighted the need for control strategies that account for battery health while ensuring that performance goals, such as speed and maneuverability, are met. In this context, Pour et al. [22] proposed a health-aware control design for autonomous racing vehicles to optimize both the system performance and battery lifespan. The proposed system integrates a Linear Parameter Varying (LPV) model to capture the nonlinear dynamics of the vehicle and a Model Predictive Control (MPC) system that maximizes the battery's RUL while minimizing lap time. This dual-objective approach aims to address both energy consumption and racing performance, thus highlighting the importance of considering battery degradation during high-speed AV operations. The approach validation was limited to a simulation environment, and further testing in real-world scenarios is necessary to assess its practicality and effectiveness. The authors emphasize that the degradation rate of batteries depends on multiple factors, including high-rate cycling and over-discharge, which can be mitigated through careful control of SOC and power consumption. By integrating these health metrics into the vehicle's decision-making process, the control system can adjust its operational parameters to balance speed and energy use, ensuring that the vehicle can complete its mission without compromising battery life. This strategy is particularly useful in high-stakes scenarios, where maintaining high performance over an extended period is crucial, and the risk of battery failure must be minimized.

Autonomous Racing and Energy Strategies

In the field of autonomous racing, energy strategies are fundamental to managing the limited resources of electric race vehicles, as energy constraints should be considered in the overall race strategy. Herrmann et al. [11] presented a real-time capable energy strategy for autonomous electric race cars that addresses the Minimum Race Time Problem (MRTP) by optimizing energy use across the entire race. This strategy incorporates the thermodynamic constraints of electric powertrain components to prevent overheating and manage battery life effectively. By solving an OC problem using a sequential quadratic programming (SQP) method, their approach ensures that race completion time is minimized without violating safety limits. This work highlights the growing importance of balancing performance with energy efficiency in autonomous racing, particularly in the context of EVs where battery capacity is a limiting factor. This strategy is crucial in minimizing total race time while preventing overheating and energy depletion, which could lead to shutdowns and premature failures during the race. The energy strategy has been validated with real-world data from the Roborace competition, although closed-loop testing is necessary to fully evaluate its performance when integrated into autonomous racing vehicles.

Battery Modeling and Performance Prediction

Accurate battery modeling is essential for predicting the performance and endurance of AVs, especially under variable and demanding conditions. Many traditional models, such as the Peukert model, are inadequate for the high discharge rates seen in such systems [12]. Advanced battery models, such as the graybox model described by Bauersfeld and Scaramuzza [12], are capable of predicting battery performance with high accuracy, even under non-constant discharge rates typical of AVs during racing or long-distance missions. Their work, although focused on aerial vehicles, provides valuable insights into how battery performance can be predicted under varying operational conditions. The range, endurance, and optimal speed for such aerial vehicles are determined through a comprehensive model that integrates blade-element-momentum (BEM) theory for multicopters with an electric-motor model, and a graybox battery model. The challenge in this area is the precise modeling of energy consumption under varying flight conditions, especially for vehicles that experience high and dynamic energy demands. By validating their model through real-world experiments, the authors demonstrate that their approach yields highly accurate predictions of battery performance, including power consumption and cell voltage under variable discharge rates. This work emphasizes the need for sophisticated models that can accurately simulate real-world conditions, particularly when aiming to maximize the efficiency of battery usage in electric-powered vehicles. While the proposed model works well for multicopters, extending its applicability to other types of vehicles or aerial systems could be a challenge.

Chapter 2

Contributions

This thesis is based on five peer-reviewed publications that address critical advancements in the fields of Li-ion battery state estimation and prediction (Sec. 2.1), and autonomous vehicles in racing contexts (Sec. 2.2). These contributions collectively advance the two fields, providing both theoretical insights and practical tools that are essential for the future of sustainable, high-performance autonomous systems.

Five open-source software repositories corresponding to the five papers have been developed as part of this research, providing the research community with access to the code and tools used in the studies. These repositories are publicly available on GitHub (see the List of Contributions index) and are accompanied by detailed documentation to facilitate their use and further development.

This research also produced two significant datasets, which are made available for the research community to encourage further research and development in the areas of battery state estimation and autonomous control. The datasets are hosted on `Mendeley Data` and `GitHub`, respectively, and are accompanied by detailed descriptions and instructions for use.

2.1 Li-ion Battery State Estimation and Prediction

In recent years, data-driven approaches utilizing DL models have revolutionized traditional battery state estimation methods. Unlike model-based approaches that rely heavily on electrochemical modeling and physical parameters, these advanced techniques leverage large datasets capturing battery performance across different conditions. This enables the development of more flexible, scalable, and accurate estimation models. The studies featured in this section emphasize the integration of neural networks such as Autoencoders, LSTMs, and Convolutional Neural Networks (CNNs), which are particularly well-suited for handling the complex, time-series data generated by batteries during charge and discharge cycles. These models have demonstrated significant improvements in the precision of SOC and RUL predictions, providing valuable insights for both academia and industry. By introducing a novel dataset and innovative model architectures, these works push the boundaries of what is possible in battery monitoring. Through the application of DL techniques, the field is moving toward more robust and generalizable solutions, enabling effective battery management even under varying operational conditions and battery aging.

2.1.1 Paper A: Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles

This paper presents a robust approach to estimating the SOC in Li-ion batteries using DL models, specifically LSTM neural networks. The research contributes to the community by introducing a new dataset, the “UNIBO Powertools Dataset”, designed for evaluating battery performance under various conditions.

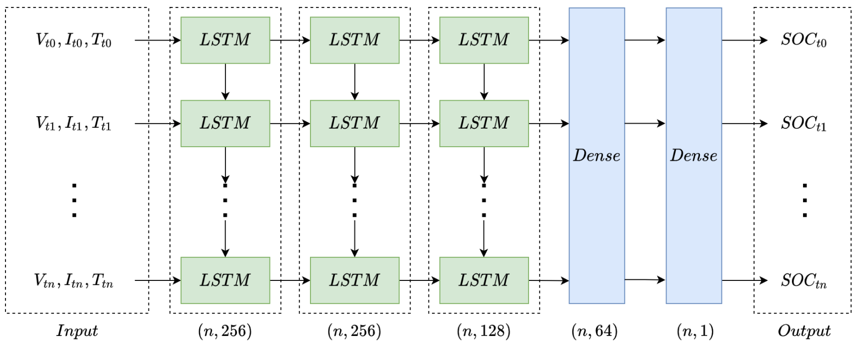


Figure 2.1: Architecture of the LSTM model used for SOC estimation on the UNIBO dataset.

Methodology

The paper employs a data-driven approach, leveraging deep LSTM neural networks to predict the SOC of Li-ion batteries based on voltage, current, and temperature data. The main innovation lies in the full use of two distinct datasets, one of which is a newly created dataset named the UNIBO Powertools Dataset. The models are trained on two datasets: the UNIBO Powertools dataset, which was specifically collected for this research, and the well-established, publicly available, LG 18650HG2 Li-ion Battery dataset. These datasets capture battery behavior across different capacities, brands, and discharge profiles, which include both CC discharges and dynamic driving profiles.

The UNIBO Powertools Dataset comprises 27 different battery cells subjected to various discharge conditions until their EOL. The dataset is unique in its inclusion of cells from different manufacturers and its coverage of battery life at different stages, providing a valuable resource for future research on how SOC is affected by battery age and health. This dataset’s diversity makes it particularly useful for training ML models to generalize better across different battery types and conditions.

The methodology centers around the use of deep LSTM networks, which are particularly well-suited for handling sequential data, such as time-series data generated by battery discharge cycles. Two distinct LSTM models are implemented: one tailored for the low-frequency sampling data from the UNIBO dataset (Fig. 2.1) and another for the high-frequency data from the LG dataset. In both cases, battery parameters like voltage, current, and temperature are fed into the models, with SOC predicted at every step. Data normalization is applied to avoid bias in training, as the input features (voltage, current, temperature) have different ranges.

Contributions

This paper makes two key contributions to the field of battery state estimation:

1. The introduction and release of the UNIBO Powertools Dataset to the public represents a significant contribution to the research community, as it offers a new and extensive resource for battery research. This dataset, publicly available, provides researchers with access to a wide range of battery types, which have different manufacturers and nominal capacities, and are cycled until EOL, enabling more robust and diverse model training for SOC, and potentially other battery management tasks like SOH and RUL. By including batteries with varying capacities, brands, and health statuses, this dataset broadens the scope of data-driven approaches in the battery domain and enhances model generalizability.

2. Additionally, the research demonstrates the capability of deep LSTM networks to accurately estimate SOC across multiple battery types and conditions. The paper presents models that are flexible enough to handle the heterogeneous nature of the datasets, including different discharge profiles and battery specifications. While the use of two models has been necessary to handle the different sampling rates, the use of two datasets highlights the models' ability to generalize across different battery technologies and environmental conditions.

Results

The models achieved a mean absolute error (MAE) as low as 0.69% on the UNIBO dataset and 1.47% on the LG dataset, with root mean square errors (RMSE) of 1.34% and 1.99%, respectively. These results show that the proposed method is effective for SOC estimation even under challenging conditions, such as dynamic driving profiles, different ambient temperatures, and varying battery degradation levels. For the UNIBO dataset, the model performed well across different discharge scenarios, with particularly strong results in the preconditioned and high-current tests. Additionally, the model was shown to be robust against changes in battery SOH, making it suitable for real-world applications where battery aging affects performance. On the LG dataset, the model was tested against different driving profiles and temperatures, showing robust performance even at lower temperatures, where battery behavior is more unpredictable.

2.1.2 Paper B: To Charge or to Sell? EV Pack Useful Life Estimation via LSTMs, CNNs, and Autoencoders

The study focuses on the challenge of estimating the RUL of Li-ion batteries, demonstrating the applicability of DL techniques for such estimation. By using autoencoders, CNNs, and LSTMs, the proposed models were able to predict the remaining Ah of batteries with high accuracy, offering a more practical solution for real-time battery health monitoring systems. The use of diverse datasets ensured the robustness of the models in handling complex real-world data with varied battery conditions.

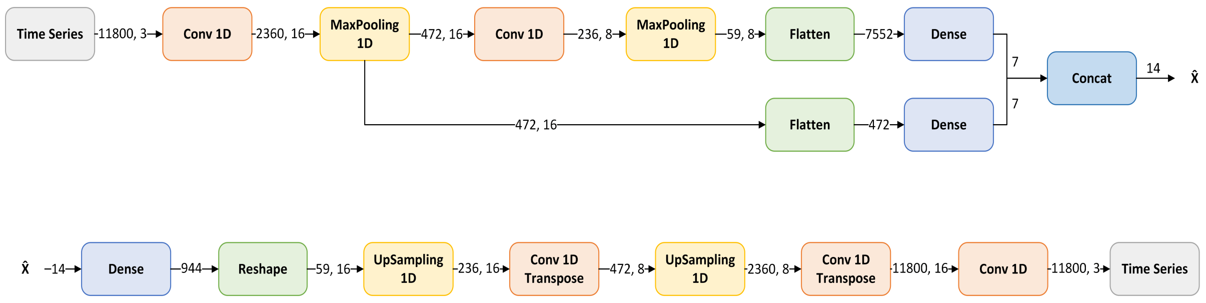


Figure 2.2: The structure of the autoencoder used to compress the cycles. The skip link allows us to retain both local and global information.

Methodology

The methodology revolves around two models for RUL prediction: autoencoder combined with CNN and autoencoder combined with LSTM. The role of the autoencoder is to extract the most relevant features from time-series data of battery discharge cycles. Afterward, CNN or LSTM is applied to estimate the battery's RUL based on these compressed representations. The autoencoder uses voltage, current, and temperature data from discharge cycles to form a compact representation of the battery's status. Its structure is shown in Fig. 2.2.

The innovation lies in predicting the remaining Ah rather than relying on the conventional cycle count for estimating the RUL, which has been traditionally used in the literature. This approach makes the method more applicable to real-world scenarios, as the remaining Ah can be more meaningful for partial charge and discharge cycles commonly seen in EV operations.

The study utilizes two large datasets: the NASA Randomized Battery Usage dataset, which contains batteries cycled under random current and various conditions, and the UNIBO Powertools dataset, which consists of batteries with a wide range of specifications cycled with different setups. These datasets offer a wide range of battery conditions and cycling patterns, providing a more realistic scenario for model generalization. The models are trained on historical data from these datasets to estimate the remaining Ah that the battery can deliver before it reaches EOL. As mentioned above, the input variables for the models include only voltage, current, and temperature data recorded during battery discharge cycles, and no future information is used in the predictions, which increases the method's applicability to real-time systems.

Contributions

This paper makes several key contributions to the field of battery health monitoring and RUL estimation, presenting:

1. A novel RUL prediction approach based on ampere-hours, which is a more practical metric in real-world EV applications, where full discharge cycles are rare and partial charge-discharge cycles are routine.
2. The use of autoencoders for efficient and robust feature extraction from large and complex high-dimensional battery time-series data, allowing the models to capture essential patterns while reducing computational complexity.
3. A comparative analysis of CNN and LSTM models for RUL estimation, providing insights into their performance in terms of accuracy, stability, and generalizability on diverse datasets.
4. Usage of datasets with wide variability. Unlike most previous work that relies on limited datasets, the full use of NASA and UNIBO datasets, which contain diverse conditions and various battery types, demonstrates the applicability of these methods to a broader range of real-world scenarios.

Results

The proposed models were evaluated on the two datasets. For the NASA dataset, the models achieved an RMSE of 0.0799 (CNN) and 0.074 (LSTM), demonstrating that both networks effectively predict the RUL defined as remaining Ah. While the CNN produced more stable predictions with well-fitted curves, the LSTM model showed better accuracy but with slightly irregular curve patterns. The reconstruction capability of the autoencoder was also highlighted, achieving an RMSE of 0.0356, which proves its effectiveness in extracting relevant features from the battery data. In the UNIBO dataset, the LSTM model achieved an RMSE of 0.021, further demonstrating its ability to learn from varied battery data. The results from both datasets indicate that the proposed methods generalize well across various batteries and operating conditions, making them suitable for real-world applications.

2.2 Autonomous Vehicles in Racing Contexts

This section delves into the application of AV technologies within the realm of racing, highlighting the research contributions from multiple papers focused on deep RL, LIDAR-based perception, and the comparison between human-piloted and OC racing systems. By examining these studies, we can better understand the advancements and limitations of current autonomous control approaches in high-speed, competitive environments and explore the implications for the future of autonomous driving. In addition to the advancements in AV techniques, the release of a novel drone racing dataset marks a significant contribution to the autonomous control community, accelerating research in autonomous racing and drone navigation.

2.2.1 Paper C: Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1^{tenth} LIDAR-Based Races

In this paper, we explored the application of deep RL in the autonomous racing domain, specifically focusing on the use of a LIDAR sensor in a small-scale autonomous car. This study addresses the challenges of transferring RL models from simulation to the real world, commonly referred to as the sim2real problem, by utilizing LIDAR data pre-processing to enhance the generalization capability of the agent. Our work integrates RL with realistic racing environments, combining practical experimentation with detailed performance evaluations.

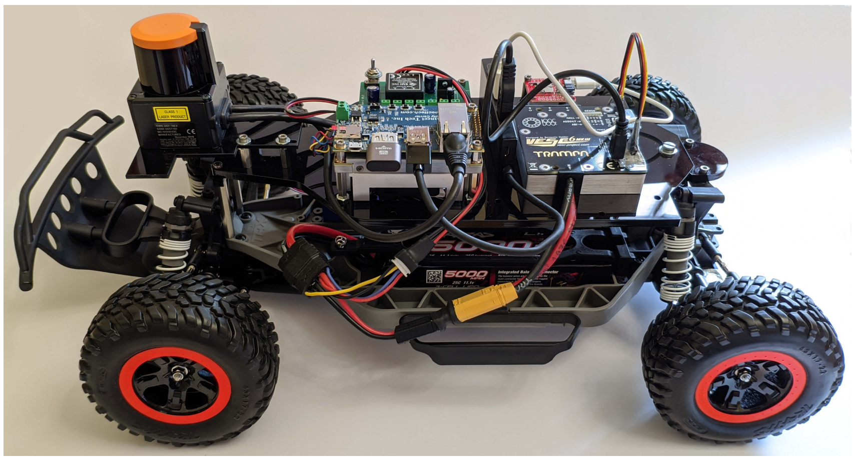


Figure 2.3: The F1tenth race car used in the experiments.

Methodology

The core methodology revolves around employing Deep Q-Networks (DQN), a popular RL algorithm, for training a 1/10 scale autonomous racing car. The study builds on the use of an F1tenth platform (Fig. 2.3), which is a realistic miniature version of Formula 1 cars designed for research purposes. This platform was equipped with a 2D LIDAR sensor, capturing a 270-degree field of view, and powered by NVIDIA's Jetson TX2 hardware for real-time data processing and training. For each experiment, the agent's actions were guided by a reward system, fine-tuned throughout the study. Initial trials used simple rewards based on movement direction, while later iterations introduced more sophisticated rewards tied to the vehicle's velocity and proximity to obstacles. The training environment implemented in ROS (Robot Operating System) coordinates sensor data, controls, and the RL agent. On top of the RL agent actions, the system's control was governed by a safety callback that automatically activated emergency braking when necessary (followed by automatic reverse), thus preventing crashes and ensuring consistent training.

We experimented with multiple neural network architectures—1D CNN, 2D CNN, and MultiLayer Perceptron (MLP)—to determine the optimal approach for processing LIDAR data in real-time autonomous driving tasks. The 1D CNN and the MLP processed the LIDAR data directly (after pre-processing), whereas, for the 2D CNN, the data were transformed into a grid map before feeding it into the network.

The system was designed to operate in both simulated and real-world environments, enabling direct comparisons of training efficacy in both domains. To address the challenges of transitioning from simulated to real-world environments, we implemented two strategies:

- Training directly in the real world using noisy LIDAR data and pre-processing techniques to filter out bad readings.
- Simulating the environment and transferring the trained model to the physical car without retraining, leveraging a robust LIDAR pre-processing mechanism to mitigate sensor noise.

The 1D CNN has been used for both sim2real experiments.

The agent was also evaluated in simulation on four complex F1 racetrack layouts, while trained only on one of them, to test its generalization capabilities across different and unseen track configurations and difficulties.

Contributions

Our contributions to the field of RL-based autonomous car racing are threefold:

1. Neural networks comparison for LIDAR data: we conducted a comparative study of different neural network architectures for processing LIDAR data, concluding that 1D CNNs provided the best performance in the autonomous racing task. The 1D CNNs outperformed both MLP networks and 2D CNNs in terms of learning speed, cumulative rewards, and overall driving performance, while also being computationally less demanding.

2. Sim2real and LIDAR sensors: we demonstrated two novel approaches to the sim2real problem. The first approach involved successfully training the car directly in the real world using noisy LIDAR data. The second approach showed that models trained in a simulated environment could be seamlessly transferred to the physical world without additional tuning, thanks to our LIDAR pre-processing pipeline. Both feats were previously considered unsolved due to LIDAR's inherent imperfections and LIDAR measurements being greatly affected by reflective surfaces [51].

3. Generalization of DQN on unseen racetracks: we evaluated the generalization abilities of DQN on complex F1 racetracks, showing that the model trained on one track was able to generalize to other tracks with varying levels of difficulty. This experiment not only validated the agent's race performance but also demonstrated its superior sample efficiency compared to both model-free and model-based RL algorithms like Dreamer.

Results

The results of our experiments show that the DQN algorithm, when paired with 1D CNNs, achieves outstanding performance in autonomous racing tasks, both in simulated and real-world settings. The 1D CNN agent learned a robust driving policy after approximately 870 training episodes (compared to 1030 and 1870 episodes of MLP and 2D CNN respectively), with significantly faster convergence and better final rewards than the other architectures tested.

In our real-world training experiments, the car learned to autonomously navigate simple tracks, demonstrating that RL-based systems can indeed be trained in noisy physical environments using LIDAR. The cumulative rewards steadily improved over three hours of training, and the car was able to handle higher speeds without requiring additional retraining.

In the transfer learning experiments, the model trained in simulation was successfully deployed on a real-world track (the rooftop of our university), with minimal performance degradation and no retraining required. The car adapted well to the real-world environment, maintaining robust control and completing laps efficiently.

Lastly, the agent demonstrated excellent generalization capabilities in F1 racetrack experiments, where it was trained on one track and tested on three unseen tracks. The DQN agent consistently outperformed baseline models, completing the race in competitive times and surpassing model-based approaches like Dreamer in both race performance and sample efficiency. The DQN model required only 550,000 steps to train, significantly fewer than the millions required by other methods, while consistently completing laps in challenging F1 tracks.

2.2.2 Paper D: Enabling Deep Reinforcement Learning Autonomous Driving by 3D-LiDAR Point Clouds

This research aims at enhancing autonomous driving through the application of deep RL techniques to 3D LIDAR point cloud data. The study aims to surpass the limitations of supervised learning approaches that rely heavily on massive labeled datasets, which are time-consuming and costly to produce. By utilizing RL, we propose a framework that autonomously learns to drive in a simulated environment thanks to a reward-based system.

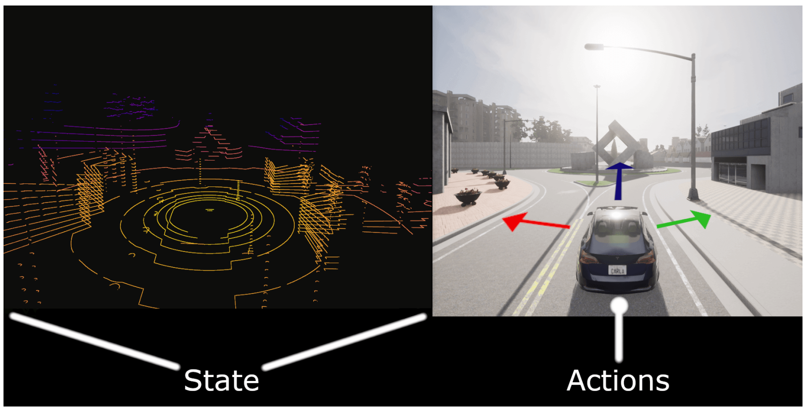


Figure 2.4: LIDAR view in CARLA with the corresponding scene from the RGB camera.

Methodology

The core methodology of this research integrates 3D LIDAR data with RL algorithms to train a simulated AV. The 3D LIDAR is a sensor that captures high-frequency point cloud data, and it is used here as the primary tool for environmental perception. The rendering of a point cloud from the CARLA simulator is shown in Fig. 2.4, left. This data is highly granular, capturing real-time spatial information in a 3D space, which helps the vehicle understand and navigate its surroundings.

The paper utilizes DQN for RL training. In this setting, the agent (the AV) learns through trial and error, guided by a reward system. The state of the agent is represented by two consecutive 3D LIDAR frames, each containing 2048 points. These points are split between an 8° vertical and 270° horizontal field of view, defining the vehicle's perception of its surroundings. The agent's actions are limited to discrete steering commands: forward, left, and right. A positive reward is given when the vehicle successfully follows the road, and negative rewards are issued when it collides or leaves the lane, resulting in the termination of the episode. This feedback loop allows the vehicle to gradually learn the optimal driving behavior.

Two neural network architectures are evaluated to process the point cloud data: PointNet and a 1D CNN. PointNet, specifically designed for point cloud data, captures the spatial relationships between points and ensures invariance to point ordering. The 1D CNN, known for its simplicity and success with 2D LIDAR data, is used as a baseline to compare performance.

The research is conducted in the CARLA simulator, which provides a hyper-realistic urban environment, as displayed in Fig. 2.4, right. CARLA allows for the simulation of various road conditions, obstacles, and pedestrian behaviors, making it an ideal platform for training and testing autonomous driving algorithms.

Contributions

This study contributes to the field of autonomous driving in several ways:

1. First application of RL to 3D LIDAR data: while RL has been applied to 2D LIDAR data and vision-based inputs, this is one of the first studies to apply RL to raw 3D LIDAR point clouds in an autonomous driving context.

2. Comparison of neural network architectures: the research evaluates two different network architectures for processing point cloud data, offering insights into their strengths and weaknesses in handling this type of data in RL tasks.

3. Simulated urban driving using LIDAR data: by leveraging CARLA, the study creates a robust simulated environment for training RL agents, providing a safe and cost-effective way to develop autonomous driving systems before transitioning to real-world scenarios.

Results

The results of the study demonstrate that both neural networks—PointNet and 1D CNN—were able to learn how to follow the street using 3D LIDAR data in the CARLA simulator. However, the 1D CNN showed slightly better performance due to its simpler structure, which was better suited for the less complex task of lane following. The average cumulative rewards across 100 episodes revealed that the RL agent had learned the basic task but still struggled with maintaining consistent performance, occasionally resulting in crashes. These findings suggest that while the approach is promising, there is still significant room for improvement, particularly in more complex driving scenarios. The loss curves during training indicate that both neural networks were able to converge, proving the networks' capacity to learn from 3D point clouds. However, further refinement is necessary for real-world deployment, as the trained agents occasionally exhibited unsafe driving behaviors.

2.2.3 Paper E: Race Against the Machine: a Fully-annotated, Open-design Dataset of Autonomous and Piloted High-speed Flight

It is presented a comprehensive dataset designed to support research and development in autonomous drone racing. The dataset is aimed at helping researchers evaluate and benchmark their algorithms in the context of fast and aggressive drone flights.

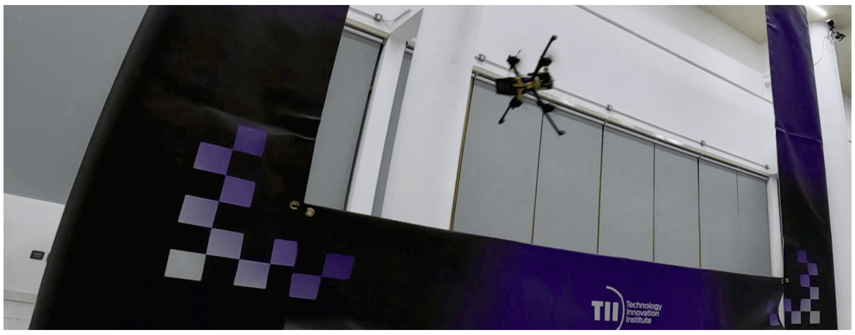


Figure 2.5: The autonomous racing drone performing an aggressive maneuver through one of the gates that form the track.

Methodology

We developed a novel dataset of autonomous and piloted drone racing flights, focusing on high-speed and aggressive maneuvers, like the one depicted in Fig 2.5. The dataset was collected using a specially designed quadrotor platform that can be used for both autonomous and piloted flights without modification. The drone's design is open-source, built with commercial off-the-shelf (COTS) components, which allows researchers to recreate the drone setup for their own experiments.

The flights were conducted in a 25 x 9.7 x 7 meters indoor arena equipped with a 32-camera Qualisys Motion Capture (MoCap) system to provide highly accurate six-degrees-of-freedom

(6DoF) poses of the drone and racing gates. The dataset includes 30 flights: 12 human-piloted and 18 autonomous, under varying illumination conditions (high, medium, and low brightness) and with different camera settings (auto exposure and fixed exposure).

The custom quadrotor features an InvenSense MPU6000 IMU for real-time tri-axis angular rate and acceleration data, along with an Arducam 8MP RGB camera for high-speed image collection at 120Hz. The quadrotor could fly at speeds up to 21.8 m/s (autonomous) through complex 3D racing tracks. Human pilots used an FPV system, while autonomous flights were performed using a Proportional Derivative (PD) controller or an MPC, depending on the flight task. Data was captured and synchronized across multiple systems, including visual, inertial, MoCap, control inputs, and battery voltage data, and processed into CSV format for easy access and use.

Contributions

The dataset contributes several unique elements to the drone racing research community:

1. Open design of the racing drone: the design of the drone used to collect the data is made publicly available, allowing researchers to recreate the platform using COTS components. This openness encourages reproducibility and standardization in drone racing research.

2. Comprehensive data collection: the dataset includes high-resolution, high-frequency visual, inertial, and MoCap data. The dataset also includes drone commands, control inputs, and battery voltages, which can be used for system identification and control-related research.

3. Release of autonomous control code along with control commands data: the dataset includes not only the flight data but also the actual control commands used by the autonomous system (PD controller for 2D tracks and MPC for 3D tracks). Furthermore, we have made the code for the PD controller publicly available, while the MPC used is a well-known open-source solution. This combined release of control algorithms and control input data provides a unique opportunity for researchers to both benchmark their own control approaches and directly replicate the control strategies used in the dataset. This transparency in control methodology enables deeper insights into the performance of various control techniques and facilitates more direct comparisons between different approaches in high-speed autonomous drone racing.

4. Versatile experimental setup: the dataset includes flights in varying lighting conditions and with different camera settings, making it a versatile tool for research in vision-based navigation and control under challenging conditions. Additionally, both autonomous and human-piloted flights are included, allowing for a direct comparison between human piloting and machine autonomy.

5. Full annotation of drone racing gates: each image frame comes with labels including gate bounding boxes and associated corners as keypoints, enabling research into scene understanding, visual-inertial odometry, and gate pose estimation.

Results

The dataset excels in several key metrics from the flights, such as top speed, acceleration, and distance covered. The autonomous flights reached top speeds of over 21.8 m/s, with the human-piloted flights reaching lower top speeds of around 9.5 m/s. The results demonstrate the drone's ability to execute aggressive and complex maneuvers, also in a challenging 3D track, making it a suitable benchmark for developing and evaluating new autonomous flight algorithms.

By providing such an extensive and well-annotated dataset, we believe our work will serve as a foundation for advancing autonomous drone racing research and creating more robust perception and control methods. The inclusion of the open-source drone design and supporting scripts further democratizes research in this area, making it accessible to a wide audience of researchers.

Chapter 3

Future Work

3.1 Integration of Battery Management and Autonomous Vehicles

The future integration of BMS and AV control represents a significant opportunity to advance the performance, efficiency, and sustainability of autonomous systems. While current research has made considerable progress in improving the efficacy of both domains separately – SOC, SOH, and RUL estimation for batteries, as well as RL and DL-based control systems for AVs – there remains a gap in fully integrating these two critical areas. The reciprocal relationship between battery management and control policies can lead to a more holistic approach where energy efficiency and vehicle performance are co-optimized.

3.1.1 Real-Time Energy-Aware Decision Making

One of the primary challenges in the integration of BMS and autonomous control is enabling real-time energy-aware decision-making. AVs, particularly in high-performance scenarios like racing or urgent delivery, require split-second decisions regarding acceleration, braking, and navigation. Currently, these decisions are often made without considering the detailed status of the vehicle's energy reserves. However, integrating BMS data, particularly SOC and SOH, into the decision-making framework can provide the vehicle with real-time insights into its energy capacity, allowing for more energy-efficient control strategies.

For instance, real-time SOC data could inform the vehicle's control policy to dynamically adjust speed and maneuvering based on available energy reserves. During a race, the vehicle might opt for more aggressive acceleration and higher speeds early on when energy reserves are plentiful, and then adopt more conservative strategies as SOC decreases, ensuring enough energy is conserved to complete the task. Similarly, in delivery or long-range missions, the vehicle could optimize its route based on real-time battery health and environmental factors such as terrain or weather conditions. This could be achieved by developing RL algorithms that incorporate battery state data as part of the observation space and energy-related incentives in the reward function, allowing the vehicle to learn energy-efficient policies that adapt to changing conditions in real-time.

3.1.2 Predictive Maintenance and Battery Lifespan Optimization

Integrating battery management with AV control also offers the potential for predictive maintenance. By continuously monitoring SOH and RUL in real-time, AVs could make informed decisions about when to reduce the load on the battery or alter operational strategies to prevent premature degradation. For example, techniques could be employed to adapt driving behavior based on battery degradation patterns over time. In scenarios where the vehicle detects an abnormal degradation rate or diminishing SOH, control algorithms could prioritize battery preservation, reducing stress on the battery and extending its RUL.

In addition to real-time decisions, predictive models for battery degradation could be developed using historical data, allowing vehicles to anticipate future maintenance needs. By incorporating ML models trained on battery usage patterns, autonomous systems could proactively schedule charging, replace worn-out components, or optimize charging strategies to extend the overall lifespan of the battery. This becomes especially important in fleet management, where downtime for maintenance can result in significant operational costs.

3.1.3 Challenges in Sim2Real Transfer and Model Generalization

One of the key challenges in achieving full integration of BMS and AV control lies in the sim2real transfer problem. While many models for battery estimation and autonomous control perform well in controlled or simulated environments, real-world conditions introduce unpredictable variations that can disrupt the performance of these systems. For example, battery behavior under variable environmental conditions—such as extreme temperatures or fluctuating power demands—can differ significantly from the assumptions made during model training.

To overcome this challenge, hybrid approaches that combine data-driven models with physical models could be employed. By leveraging the strengths of both methods, vehicles could generalize better across different operational environments. Furthermore, ongoing advancements in transfer learning could enable models trained in one environment (e.g., controlled laboratory settings) to adapt and function in real-world applications with minimal additional training.

3.1.4 Toward a Unified Framework

The future of autonomous systems lies in the development of a unified framework that seamlessly integrates battery management and autonomous control. Such a framework would require collaboration between various domains, including ML, electrical engineering, and control theory. By designing systems where control policies and battery health monitoring work together, the performance and longevity of AVs could be greatly enhanced.

This integration will be particularly important as the demand for high-performance, energy-efficient autonomous systems continues to grow in industries such as logistics, transportation, and emergency response. Autonomous drones, for example, could benefit from this integration by optimizing flight paths to maximize range and ensure safe landing before battery depletion. In electric autonomous cars, the same framework could reduce overall operational costs and improve vehicle longevity, making autonomous technology more economically viable and environmentally sustainable.

In conclusion, the integration of BMS with AV control is an exciting frontier that promises to elevate the capabilities of autonomous systems. By combining real-time battery health data with adaptive control strategies, future AVs will not only be more efficient but also more resilient and capable of performing complex tasks under varying operational conditions. The path forward requires ongoing research in ML, system modeling, and the development of reliable, generalizable models that can operate seamlessly across both simulated and real-world environments.

Appendix A

Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles

Reprinted, with permission, from:

K. L. Wong, M. Bosello, R. Tse, *et al.*, "Li-ion batteries state-of-charge estimation using deep lstm at various battery specifications and discharge cycles," in *Proceedings of the Conference on Information Technology for Social Good*, ser. GoodIT '21, ACM, 2021, pp. 85–90. DOI: 10.1145/3462203.3475878

Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles

Kei Long Wong, Michael Bosello, Rita Tse, Carlo Falcomer, Claudio Rossi, Giovanni Pau

Abstract – Lithium-ion battery technologies play a key role in transforming the economy reducing its dependency on fossil fuels. Transportation, manufacturing, and services are being electrified. The European Commission predicts that in Europe everything that can be electrified will be electrified within a decade. The ability to accurate state of charge (SOC) estimation is crucial to ensure the safety of the operation of battery-powered electric devices and to guide users taking behaviors that can extend battery life and re-usability. In this paper, we investigate how machine learning models can predict the SOC of cylindrical Li-Ion batteries considering a variety of cells under different charge-discharge cycles.

A.1 Introduction

The transition from fossil fuel to green energy is well known as the desired change in our society. To reduce the emission of Carbon dioxide (CO_2) from conventional transportation, the development of Electric Vehicles (EV) is growing quickly. Battery technology will be one of the most important key enablers for the green energy transition.

Lithium-ion batteries have been widely used in electric vehicles. It is projected that the global EV stock will expand to 140 million by 2030 [52]. Lithium-ion (Li-ion) battery is the most popular adopted power supply of EV due to its high energy density, long lifespan, lightweight, and low self-discharge rate [53]. Several factors could affect the performance and safety of Li-ion battery such as ambient temperature, **over-charge**, or **over-discharge** [54], [55]. A misuse of the battery can lead to a **shorter** battery life. To overcome these issues, Battery Management Systems (BMS) are applied to ensure the reliability and stability of the usage of Li-ion batteries.

One important parameter for the BMS battery health management is the battery State Of Charge (SOC) estimation which helps to prevent the battery from over-charge and over-discharge [8], [56]. SOC indicates the amount of available charge in the battery which can be represented by a value in percentage. This value is intended to remain between 0% and 100%, although it is possible to violate these limits in an over-discharge or over-charge situation [24]. The battery itself does not directly provide information on its SOC value. The measurement of SOC value is complex and error-prone due to the indirect estimation and the non-linear nature of electrochemical reactions in the battery. Relevant information such as the measured discharge current, voltage, and ambient temperature can be used to measure the SOC indirectly [23].

Incorrect measurement of SOC could lead to unstable EV performance and even shorten the battery life, therefore, reducing the environmental benefits of electrification. In general, the SOC estimation techniques studied in the literature can be divided into three categories: direct methods, model-based methods, and data-driven methods [25]. The direct methods look for the relationship between SOC and the physical battery characteristic parameters. The SOC value can be estimated according to the observed parameters[24],[25], [57].

The model-based SOC estimation methods mainly focus on modeling the chemical and electrical properties of the battery.

Commonly, the model-based methods are used in collaboration with adaptive filters such as Kalman filter, H-infinity filter, and Particle filter, etc [25], [58]. Model-based methods require a comprehensive understanding of the electrochemical properties in the battery domain and cannot be used for SOC forecast[59][60].

This work proposes a data-driven approach for SOC estimation based on Deep Learning techniques. Deep learning, which can approximate non-linear functions, is a widely adopted data-driven method to tackle the battery SOC estimation problem [13]. Given a sufficient amount of training data and an appropriate configuration, the SOC value can be predicted accurately without the need for a sophisticated electrochemical model. Different types of neural networks

(NNs) such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been studied in the literature to solve various problems of different nature [61], [62]. Among them, RNNs are designed to handle sequential data, and they have been well studied in the domain of speech recognition and natural language processing with successful outcomes [63], [64]. However, RNNs struggle to handle long-term dependencies as long time series could cause exploding/vanishing gradient during the training phase. To tackle this problem, [26] proposed the use of RNNs with Long Short-Term Memory (LSTM) cells which can correlate a long-range of precedent information.

In the literature, various applications of LSTM for SOC estimation have been proposed. [27] showcased the ability of LSTM for SOC estimation. They used a dataset composed of discharge cycles obtained from a Li-ion battery cell with 2.9Ah nominal capacity [65] and collected through laboratory testing under different driving profiles. The proposed model was validated against various ambient temperatures with accurate estimation results achieved. Similarly, [66] proposed the use of a deep LSTM network with data collected from an experiment on a 1.1Ah nominal capacity battery cell. In [67], a neural network combining CNN and LSTM layers was proposed for battery SOC estimation. The research result has shown that the CNN part helps to extract the spatial features from the input data (voltage, current, and temperature) while the LSTM layers explore the correlation of current SOC and historical input data. Last but not least, the use of an LSTM encoder-decoder algorithm was proposed by [68] with accurate estimation result against both room temperature and various temperature conditions. The proposed model was trained and tested on 2.0Ah nominal capacity Li-ion battery cell data featuring various drive cycles.

Although the neural network data-driven approaches for SOC estimation are widely studied most of the literature mainly focuses on datasets containing only **one particular battery model** or setup. This study uses two different Li-ion cell datasets. The first one is original and it has been collected by the University of Bologna (UNIBO), namely the 'UNIBO Powertools Dataset'. The second one is public, the LG 18650HG2 Li-ion battery data [69]. The use of deep LSTM networks is proposed to perform SOC estimation. Due to the heterogeneity of the data collection process and the sampling rate of the two datasets, two deep LSTM models with different setups are employed in this research. The deep networks are tested on Li-ion battery cells with different nominal capacities, specifications, and brands; the discharge cycles are produced by both constant current discharge and several dynamic driving profiles (such as the Urban dynamometer driving schedule (UDDS) [70]).

The rest of this paper is organized as follows. The employed battery datasets are introduced in section A.2. Then, section A.3 explains the proposed deep LSTM models. In section A.4, the results of the experiments are presented. Finally, section A.5 concludes this paper.

A.2 Li-ion battery datasets

In this paper, two Li-ion battery datasets – one original and one public – with different features are used. The UNIBO Powertools Dataset is presented here for the first time, and it is available here[1]. Only the discharge cycles are used in the experiments. The two datasets are briefly introduced in the following sections.

A.2.1 UNIBO Powertools Dataset

The UNIBO Powertools Dataset has been collected in a laboratory test by an Italian Equipment producer. The cycling experiments are designed to analyze different cells intended for use in various cleaning equipment such as vacuum and automated floor cleaners. The vast dataset is composed of 27 batteries, and it is summarized in Table A.1. The main features of the dataset are: (1) the use of batteries from different manufacturers, (2) cells with several nominal capacities, (3) cycling is performed until the cell's *end of life* and thus data regarding the cell at different life stages are produced, which is useful to assess **how SOC is affected by the cell's age** and State of Health (SOH) as well as to validate the capability of the proposed model on estimating SOC under different health status. Three types of tests have been conducted. (I) The **standard test**, where the battery was discharged at 5A current in main cycles. (II), the **high current test**, where the battery was discharged at 8A current in main cycles. (III), the **preconditioned test**, where the battery cells are stored at 45°C environment for 90 days before conducting the test.

During discharge, the sampling period is 10 seconds. The experiments were conducted using the following procedure:

1. Charge cycle: Constant Current-Constant Voltage (CC-CV) at 1.8A and 4.2V (100mA cut-off)
2. Discharge cycle: Constant Current until cut-off voltage (2.5V)
3. Repeat steps 1 and 2 (main cycle) 100 times
4. Capacity measurement: charge CC-CV 1A 4.2V (100mA cut-off) and discharge CC 0.1A 2.5V
5. Repeat the above steps until the end of life of the battery cell

Table A.1: UNIBO Powertools Dataset summary

Test type	Nominal capacity	Cell amount
Standard	4.0Ah	2
	3.0Ah	4
	2.85Ah	4
	2.0Ah	6
High current	3.0Ah	3
	2.85Ah	2
Preconditioned	3.0Ah	5

[1]https://doi.org/10.17632/n6xg5fzsbv.1

A.2.2 LG 18650HG2 Li-ion Battery Data

The public LG 18650HG2 Li-ion Battery Dataset, published by [69], was obtained from Mendeley data. In the dataset, a series of tests were performed under six different temperatures. The battery was charged at 1C rate to 4.2V with 50mA cut off before each discharge test. The values measured in the discharge cycles are captured at 0.1 seconds sampling period. Different drive cycles such as UDDS, LA92, and US06, as well as mixes of them, are applied in the discharge tests. In this paper, the discharge cycles with temperature of 0°C, 10°C and 25°C were used for training and testing the proposed model.

A.3 Methodology

In this section, the basic theories of RNN and LSTM are introduced and the two proposed deep LSTM models are briefly introduced. Then, the normalization method used to scale the input data is reviewed. Lastly, the model's configuration is discussed.

A.3.1 Recurrent Neural Networks Primer

Recurrent neural networks are a class of neural networks that allows the information to persist over time. Different from the feed-forward neural networks that are acyclic directed graphs, RNNs have connections within layers forming cyclic directed graphs. This empowers neural networks to have a state, and thus memory. The information from the previous state is utilized as input along with the current time step. It is useful for sequential data prediction as relationships between current and past information are considered. An example of the architecture of an RNN for SOC estimation unfolded in time, is depicted in Fig. A.1. The input vector at the time step *t* contains battery parameters such as voltage, current, and temperature, and it is denoted as $Input_t$. h_t represents the hidden state at time step *t*, while the output SOC value at time step *t* is denoted as SOC_t. Fig. A.1 demonstrates a common approach for time-series called many-to-many, where multiple input steps are fed to the network with one prediction made at each step. Whereas, there are other approaches such as the many-to-one and one-to-many, where in the first case multiple time-steps are fed with one output produced, and in the second case one input is used to produce multiple time-steps. As the two battery datasets have very different sampling frequencies, we used the many-to-many approach for the first model (low-frequency sampling) while in the second one (high-frequency sampling) we used the many-to-one approach.

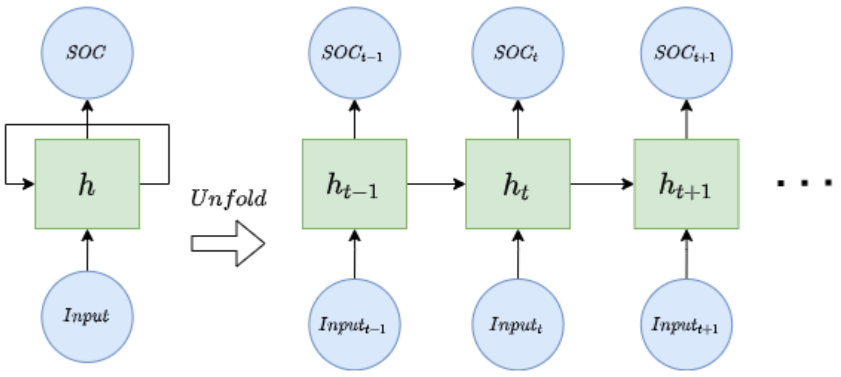


Figure A.1: RNN architecture for SOC estimation unfolded in time

A.3.2 Long Short-Term Memory Primer

The long short-term memory is a type of RNN which is widely used to learn long-term dependencies without experiencing the exploding and vanishing gradient problems. The forward pass of an LSTM cell can be defined by the following steps. In the equations, f_t, i_t, o_t are the forget-gate, input-gate, output-gate; c_t and h_t are the cell state and hidden state at time step t respectively; σ is the sigmoid function; $\odot$ is the Hadamard product; W denotes the weight matrix; x_t is the input vector at time step t and b is the bias.

The first step in the LSTM cell is to determine what information will be forgotten from the cell state c_{t-1}. The forget-gate uses a sigmoid function, in which outputs are always between 0 and 1. The result represents therefore how much should be forgotten, with 0 and 1 representing respectively discarding everything or keeping everything from the previous cell's state. As shown in the equations, the decisions of gates are based on the current input and hidden state as well as on the network's weights and biases.

$$f_t = \sigma(W_x^f x_t + W_h^f h_{t-1} + b^f) \tag{A.1}$$

The second step determines whether the information will be stored in the cell state. There are two parts in the second step. Firstly, the input-gate with sigmoid output determines to what extent the value will be remembered. Secondly, the tanh layer generates the new value $\tilde{c}_t$ that is multiplied by the sigmoid output and then added to the cell state.

$$\begin{aligned} i_t &= \sigma(W_x^i x_t + W_h^i h_{t-1} + b^i) \\ \tilde{c}_t &= tanh(W_x^c x_t + W_h^c h_{t-1} + b^c) \end{aligned} \tag{A.2}$$

The cell state c_t is then update combining the previous cell state c_{t-1} with new value $\tilde{c}_t$ as mentioned above. The forget-gate f_t and input-gate i_t determine whether the values should be discarded or remembered.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{A.3}$$

In the last step, the output-gate with the sigmoid function decides which part of the cell state is propagated to the hidden state h_t. In the hidden state, the cell state c_t is passed via $tanh$ and multiplied by the output-gate to keep only the desired output.

$$\begin{aligned} o_t &= \sigma(W_x^o x_t + W_h^o h_{t-1} + b^o) \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{A.4}$$

A.3.3 Proposed LSTM Approach

There are two deep LSTM models proposed in this paper, one for each dataset, as they have very different cycle lengths. Scaled exponential linear units (SELU) [71] activation function is used in all the LSTM cells and hidden dense layers. In the output layer, the linear activation function is applied to produce the final SOC value.

The first model is used for the UNIBO dataset. It is a deep neural network with three LSTM layers followed by two dense layers to map the learned states to desired SOC output. The number of cells of each LSTM layer is 256, 256, and 128 respectively. Fig. A.2 illustrates the architecture of the first proposed model. The first layer is the input layer with battery parameters including voltage V, current I, and temperature T at each step t. Since it is a deep LSTM network, each LSTM layer returns a sequence which means that each step is propagated to the next layer. Here, we adopted the many-to-many approach, the SOC value is therefore estimated at every step. The input time series fed to the deep LSTM network is defined as $[Input_{t0}, Input_{t1}, ...Input_{tn}]$, where n is the number of steps in the entire discharge cycle, and $Input = [V_t, I_t, T_t]$ represents voltage, current and temperature at each time step respectively. Although the entire discharge cycle is fed to the network, only the part that precedes the step under examination is available as input for SOC estimation, i.e., the hidden state from previous steps $t-1$ and the current input at step t are used to estimate the output at step t.

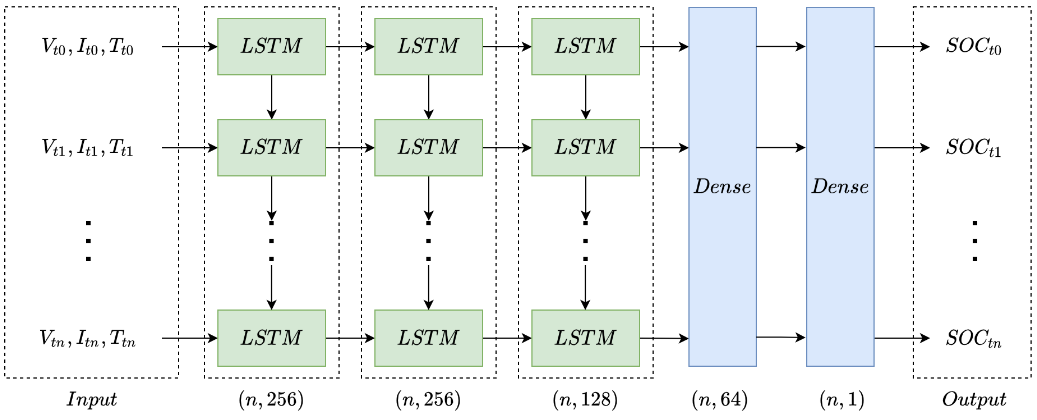


Figure A.2: Architecture of the first model

The second model is used for the LG 18650HG2 Li-ion battery dataset. The model is composed of two LSTM layers followed by three dense layers. The number of cells of both LSTM layers is 256. Fig. A.3 shows the architecture of the second proposed model. Since the second dataset contains more steps in one discharge cycle due to its higher sampling rate (0.1 seconds sampling time), the many-to-one approach is more appropriate. In this case, for each n step as input, one output is returned. In the implementation, we used 300, 500, and 700 as the number of steps. For example, given input steps $[Input_{t0}, Input_{t1}, ...Input_{t500}]$, the model should estimate the SOC value at step 500.

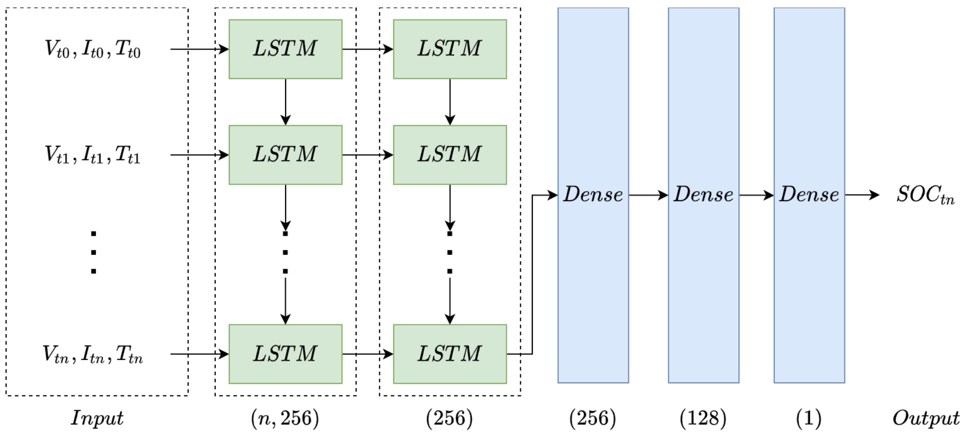


Figure A.3: Architecture of the second model

A.3.4 Data Normalization

Since the input features have different ranges, such as the temperature has much higher values than voltage and current, the trained model could give more importance to this feature over the others due to its larger value. To avoid this problem, the minimum-maximum normalization method is used to scale all input features into the same common scale.

A.3.5 Model Training

The proposed models are implemented by using the Keras library [72]. The Adam algorithm [73] is chosen as the optimizer to update the network weights and biases with the learning rate configured as 0.00001. All proposed models are trained for 1000 epochs, but the training process would stop earlier if there is no further improvement of validation loss within 50 epochs. The Huber loss [74] is used as the loss function. Its peculiarity is that it can be quadratic or linear depending on the error value.

A.4 Results and discussion

The proposed deep LSTM models are trained and tested using the two aforementioned datasets. The model performance against each dataset is discussed in this section. The source code of the model implementation and results are available here[2].

Root Mean Square Error (RMSE) and Mean absolute error (MAE) are used to evaluate the proposed models. The Mean Square Error (MSE) is the sum of squared distances between the target and predicted variables divided by the number of samples. The RMSE is the square root of the MSE which scales the output value to the same scale as MAE. It is more sensitive to outliers as it penalizes the model by squaring the error. The MAE on the other hand is more robust to outliers as the error is not squared. MAE is an L1 loss function that calculates the sum of the absolute difference between the target and predicted variables. The MAE is more suitable for problems where the training data present outliers.

A.4.1 UNIBO Powertools Dataset

In the UNIBO dataset tests, the performance of the proposed model is evaluated over constant current discharge. The proposed model for this dataset was trained with a total of 7738 discharging cycles as the training set. One cell for each group of test types (standard, high current, pre-conditioned) and nominal capacity was extracted as testing data for evaluation purposes. The testing data is isolated from training data so that it is unseen during the training process. The overall MAE and RMSE on all testing data are 0.69% and 1.34% respectively.

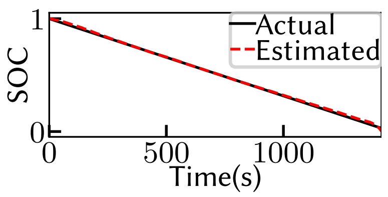
To further investigate the performance of the proposed model, Table A.2 shows the performance of each test type. The evaluation of standard test type with 4.0Ah nominal capacity and high current test type with 2.85Ah nominal capacity has the worst performance. This is expected as the dataset contains only two cell tests of the kind, resulting in one cell used for training and one for testing. Whereas, in the other test types with sufficient data the model can achieve accurate results with RMSE lower than 1%.

[2]https://github.com/KeiLongW/battery-state-estimation

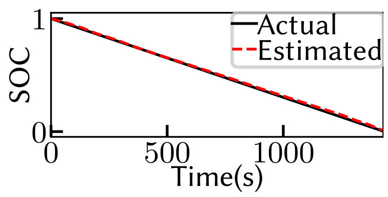
Table A.2: UNIBO dataset tests performance

Test type	Nominal capacity	MAE	RMSE
Standard	4.0Ah	2.68%	3.42%
	3.0Ah	0.52%	0.73%
	2.85Ah	0.31%	0.39%
	2.0Ah	0.59%	0.80%
High current	3.0Ah	0.46%	0.61%
	2.85Ah	2.13%	3.24%
Preconditioned	3.0Ah	0.47%	0.66%

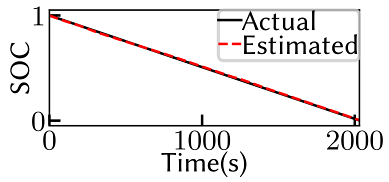
The examples of SOC estimation results of the proposed model on the standard, high current, and preconditioned test types are shown in Fig. A.4, Fig. A.5, and Fig. A.6 respectively. The first and the last discharge cycles within the entire test of each battery cell are presented to demonstrate the SOC estimation performance under different health statuses. All results show the discharge process of SOC being discharged from 100% to 0%. The x-axis represents the discharge steps over the whole discharging cycle and the y-axis represents the SOC value at each step. The black line is the actual observed SOC value during the discharge process and the red dashed line is the SOC value estimated by the proposed model.



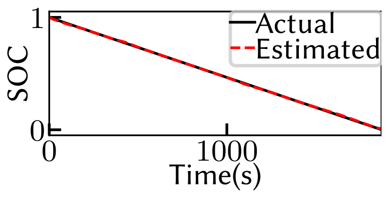
(a) Standard 2.0Ah (first cycle)



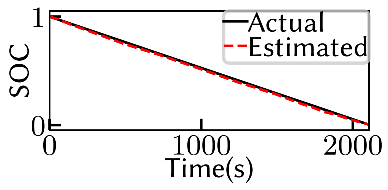
(b) Standard 2.0Ah (last cycle)



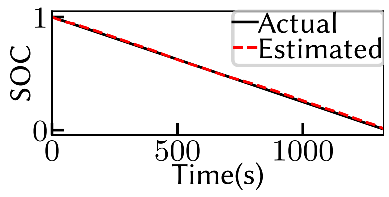
(c) Standard 2.85Ah (first cycle)



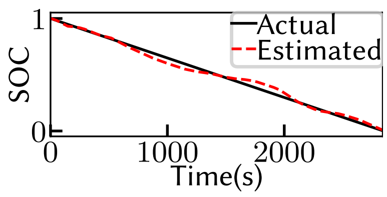
(d) Standard 2.85Ah (last cycle)



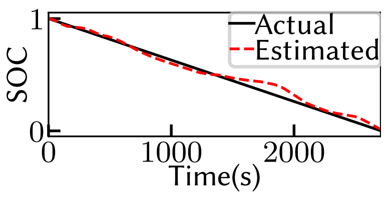
(e) Standard 3.0Ah (first cycle)



(f) Standard 3.0Ah (last cycle)

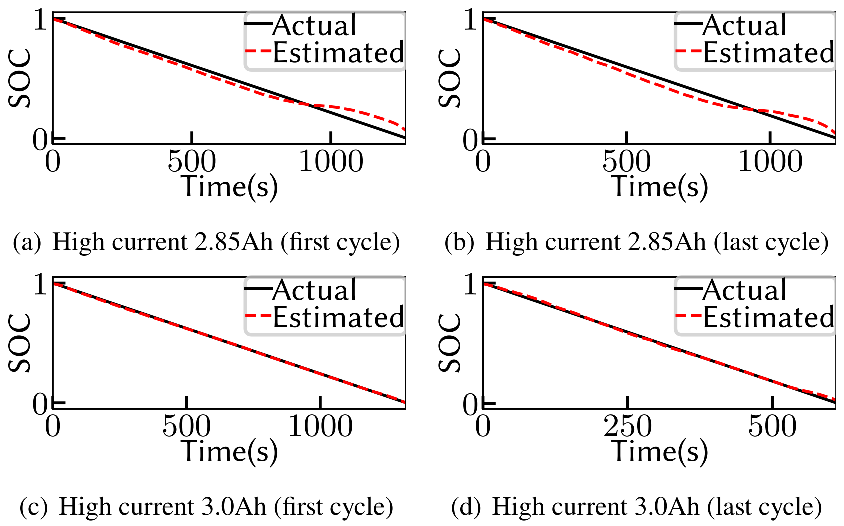


(g) Standard 4.0Ah (first cycle)



(h) Standard 4.0Ah (last cycle)

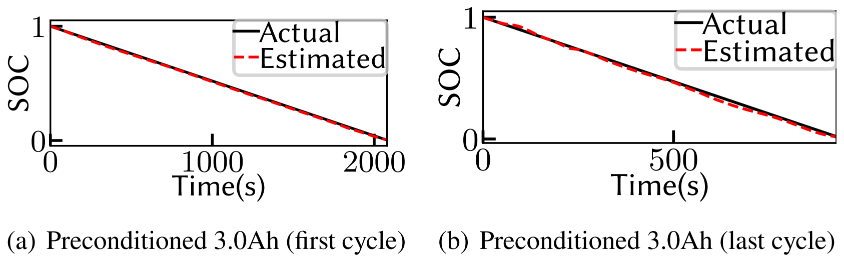
Figure A.4: UNIBO dataset SOC estimation results (standard)



(a) High current 2.85Ah (first cycle) (b) High current 2.85Ah (last cycle)

(c) High current 3.0Ah (first cycle) (d) High current 3.0Ah (last cycle)

Figure A.5: UNIBO dataset SOC estimation results (high current)



(a) Preconditioned 3.0Ah (first cycle) (b) Preconditioned 3.0Ah (last cycle)

Figure A.6: UNIBO dataset SOC estimation results (preconditioned)

The model estimates the SOC of the 3.0Ah nominal capacity cells correctly and without large fluctuation in each of the three test types. Furthermore, the estimations of standard test types with 2.0Ah and 2.85Ah nominal capacity are accurate too. SOC in both the first and last cycle are estimated accurately which suggests that the proposed model is capable to estimate SOC under different battery health statuses. In addition, good performance is achieved from the preconditioned test type which demonstrates that the storage temperature before testing does not affect the battery discharging behavior significantly in terms of SOC estimation. On the other hand, there are some errors during the ending steps of standard 4.0Ah nominal capacity and high current 2.85Ah nominal capacity battery cell cycles. It is acceptable as there is only one training example of that kind of setup.

A.4.2 LG 18650HG2 Li-ion Battery Data

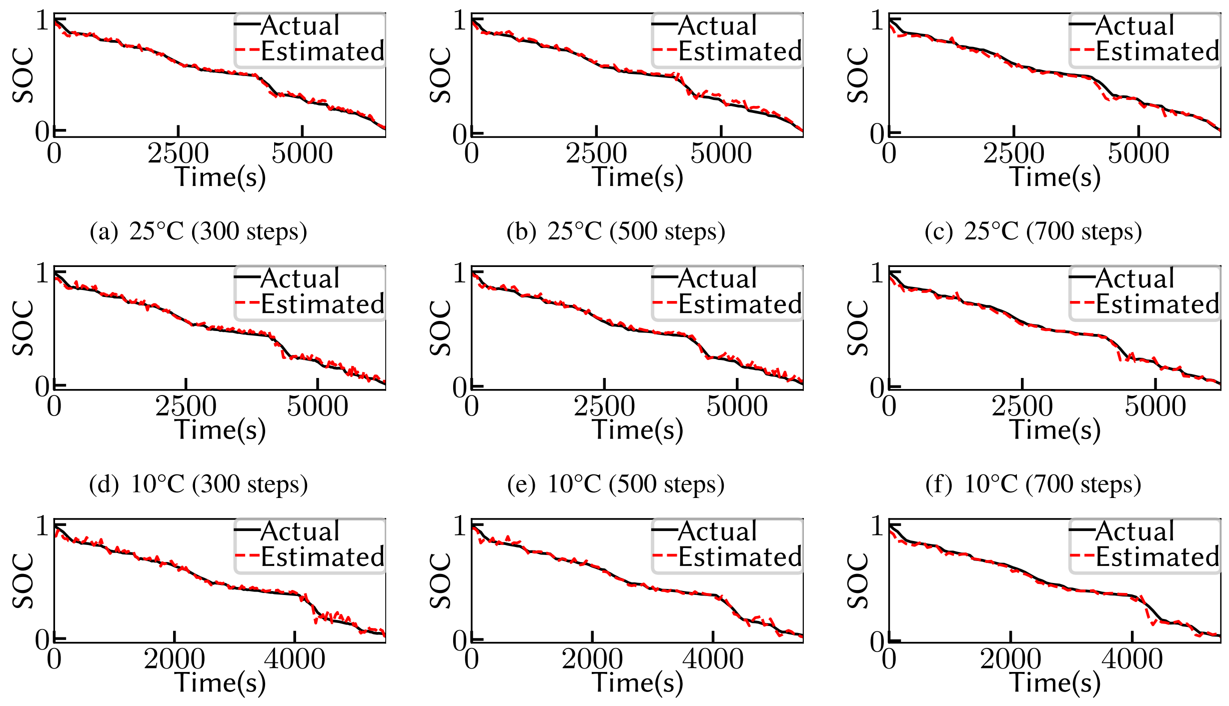
In the LG 18650HG2 Li-ion battery dataset, the performance of the proposed model under dynamic discharge current is evaluated. Six mixed driving cycles for each of three different temperatures 0°C, 10°C, and 25°C were used as training set. We have also tested three different time series lengths, with a number of steps of 300, 500, and 700, which are approximately equal to 30 seconds, 50 seconds, and 70 seconds depth in time respectively. The test set was composed of a UDDS, an LA92, and a US06 driving cycle plus one mixed driving cycle for each of the three different temperatures available in the dataset.

The MAE and RMSE achieved by the 300 steps model are 1.47% and 1.99%. The 500 steps one reached an MAE and RMSE of 1.54% and 2.12%. The 700 steps model achieved 1.94% MAE and 2.72%

RMSE. All the aforementioned results were tested with testing data under all temperatures. The model performance under each temperature with different input lengths is listed in Table A.3. Among all the configurations, the best performance is achieved from testing data under 25°C temperature with 300 steps in input, which demonstrates that the battery operates most stably under room temperature. The model is able to learn the battery behavior under room temperature through the provided driving cycles without the need for a long history. While, under 10°C and 0°C temperatures, better performance is gained from the 500 input model. This indicates that increasing input steps could help to improve the estimation result under temperatures that are lower than room temperature. However, the worst results are from the 700 input steps which suggest that the increment of input steps must be selected carefully for the many-to-one approach as an inappropriate increment of input steps could result in performance degradation. The SOC estimation results on the mixed driving cycles under 0°C, 10°C and 25°C temperatures are displayed in Fig. A.7. The estimation results under the three temperatures are competitive and without significant errors. Still, errors can be seen from the ending steps in mixed cycles under 0°C temperature due to their more dynamic discharge pattern.

Table A.3: LG 18650HG2 data tests performance

Temp. (°C)	300 Steps		500 Steps		700 Steps	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
0	1.69%	2.27%	**1.47%**	**2.23%**	1.65%	2.60%
10	1.61%	2.12%	**1.57%**	**2.12%**	2.22%	2.89%
25	**1.17%**	**1.57%**	1.59%	2.02%	1.92%	2.64%



(a) 25°C (300 steps) (b) 25°C (500 steps) (c) 25°C (700 steps)

(d) 10°C (300 steps) (e) 10°C (500 steps) (f) 10°C (700 steps)

(g) 0°C (300 steps) (h) 0°C (500 steps) (i) 0°C (700 steps)

Figure A.7: LG 18650HG data SOC estimation results (mixed cycles)

A.5 Final remarks

In this paper, a deep LSTM NN is proposed to estimate SOC over two different Li-ion battery datasets. Discharge cycles with both constant and dynamic current under various ambient temperatures are used to train and test the proposed models. The evaluation results show that the proposed models can learn the battery dynamic behavior during discharge. Battery SOC can be estimated accurately by using the measured voltage, current, and temperature values, with 1.34% and 1.99% RMSE in constant current and dynamic current discharge cycle respectively. We have also shown how the proposed estimation is robust w.r.t. different State of Health statuses. The SOH is another important parameter for battery management. As future work, we suggest using deep LSTM networks for SOH estimation as we believe it can be effective as well.

Appendix B

To Charge or to Sell? EV Pack Useful Life Estimation via LSTMs, CNNs, and Autoencoders

Reprinted, with permission, from:

M. Bosello, C. Falcomer, C. Rossi, *et al.*, "To charge or to sell? ev pack useful life estimation via lstms, cnns, and autoencoders," *Energies*, vol. 16, no. 6, 2023. DOI: 10.3390/en16062837

To Charge or to Sell? EV Pack Useful Life Estimation via LSTMs, CNNs, and Autoencoders

Michael Bosello, Carlo Falcomer, Claudio Rossi and Giovanni Pau

Abstract – Electric vehicles (EVs) are spreading fast as they promise to provide better performance and comfort, but above all, to help face climate change. Despite their success, their cost is still a challenge. Lithium-ion batteries are one of the most expensive EV components, and have become the standard for energy storage in various applications. Precisely estimating the remaining useful life (RUL) of battery packs can encourage their reuse and thus help to reduce the cost of EVs and improve sustainability. A correct RUL estimation can be used to quantify the residual market value of the battery pack. The customer can then decide to sell the battery when it still has a value, i.e., before it exceeds the end of life of the target application, so it can still be reused in a second domain without compromising safety and reliability. This paper proposes and compares two deep learning approaches to estimate the RUL of Li-ion batteries: LSTM and autoencoders vs. CNN and autoencoders. The autoencoders are used to extract useful features, while the subsequent network is then used to estimate the RUL. Compared to what has been proposed so far in the literature, we employ measures to ensure the method's applicability in the actual deployed application. Such measures include (1) avoiding using non-measurable variables as input, (2) employing appropriate datasets with wide variability and different conditions, and (3) predicting the remaining ampere-hours instead of the number of cycles. The results show that the proposed methods can generalize on datasets consisting of numerous batteries with high variance.

B.1 Introduction and Background

Electric vehicles (EVs) are becoming central to the automotive industry as they can address current automotive limits. Their constant growth is due to their improved performance and efficiency, but especially for their suitability in addressing environmental challenges, i.e., urban pollution and global warming [75], [76]. Internal-combustion-based vehicles contribute to global carbon emissions by 14% of the total [77]; thus, they are facing restrictions in leading markets that aim to reduce their environmental footprint [76], [78]. Internal-combustion-based vehicles are also a prominent source of artificial fine particulate matter ($PM_{2.5}$) [79], [80]. Air pollution is one of our greatest social issues since it has a severe impact on health and society [81], possibly causing different diseases and even premature death [82], [83]. EVs are a milestone in addressing such challenges to humanity as they can potentially remove personal transportation from the environmental impact equation.

A core component of EVs is the battery. Lithium-ion (Li-ion) batteries have become the standard for energy storage in EVs [18], [19]. They have several advantages compared to traditional batteries such as lead-acid or nickel-metal hydride: high energy and power density, low self-discharge, environmental adaptability, long lifetimes, and high reliability [8], [84]. These advantages have led to the wide use of Li-ion batteries in EVs and in several safety-critical areas such as space applications [85], aircraft, and backup energy systems. The safety and reliability of Li-ion batteries are critical concerns for such applications [86]. Li-ion batteries are employed in safety-critical areas, so their defects can cause fatal system failures. For example, various Boeing 787 aircraft caught fire because of Li-ion battery malfunctions in 2013 [15], and NASA lost a spacecraft because of the lack of power supply due to a false battery over-charging indication in 2006 [16]. Such high-impact failures have also recently appeared in the EV domain, with well-known manufacturers recalling hundreds of thousands of EVs due to fire risk [87], [88]. Another far more significant challenge for Li-ion batteries is their cost. While EVs are promising on various fronts, their cost is still a considerable drawback [89], and the battery is one of the most expensive components of EVs [90].

The design of an appropriate battery management system (BMS) is crucial to reducing costs and increasing vehicle efficiency and security [6], [7]. One of the major tasks of the BMS is to evaluate the current health conditions of the battery as it degrades over time. This degradation is an irreversible process related to the repetitive charging and discharging operations and electrochemical reactions inside the battery [21]. Predominant indicators are battery capacity and internal resistance, which inform us about the battery residual energy and power capabilities, respectively [28], indicated by the state of health (SOH). The SOH and the remaining useful life (RUL) are the most crucial parameters of battery health that must be estimated by the BMS [8]. The SOH quantifies the deterioration level compared to a brand new battery. While it has not been formally defined by industry [91], it is typically expressed through a percentage of capacity loss or power loss (increase in battery resistance) [13], [92]. We will consider the capacity loss (SOHc), which is defined by

$$SOH = \frac{C_t}{C_0} \cdot 100(\%) \tag{B.1}$$

where C_t is the current capacity and C_0 is the nominal capacity. The International Electrotechnical Commission (IEC) [93], International Organization for Standardization (ISO) [94], and Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA) [95] have proposed standards to measure the battery capacity in a standard condition using direct methods that are taken as a reference to compute C_t and C_0. The BMS adjusts its functioning according to the estimated SOH to ensure vehicle performance and safety until the health indicators reach the target limits, after which the battery should be replaced. Battery manufacturers usually set the capacity threshold under which the battery is no longer suitable for EV applications to 80% of the nominal capacity [96], measured under a standardized test. Such threshold is called the end-of-life (EOL). Despite this, the battery might be replaced before the threshold if the internal resistance rises above a normal level [28]. The threshold is also recommended to be 80% by the Center for Advanced Life Cycle Engineering (CALCE) at the University of Maryland [97] and 70% by NASA's Prognostics Center of Excellence (PCoE) [34]. In the context of replacement and secondary use planning, it is helpful to *predict* how the SOH will evolve through time and when the battery will reach its EOL. This is defined by the RUL, which is typically described as the number of cycles remaining until EOL [30]–[32].

A robust SOH estimation by the BMS is fundamental to ensure battery reliability as well as prevent failures and hazards [6], [98], but also to determine the acceleration performance and the driving range of the EV [20], [99], necessary for a pleasant driving experience, and finally to quantify the residual market value of the batteries [100]. However, the correct estimation of the RUL encourages the reuse of batteries, as removing the battery before it exceeds the end of life of the target application allows reuse in a second domain without compromising safety and reliability [29]. Batteries can therefore be employed in secondary applications with lower power requirements. This can have a significant impact in terms of both sustainability and market value [101]. With the growing number of new EVs, the waste produced by the spent battery packages is also increasing. Their recycling processes can have a considerable economic and environmental impact. The interested reader is referred to [102] for lithium-ion battery recycling in the EV context. To summarize, improving the estimation of SOH and RUL contributes to the spreading of EVs in two ways: (1) by ensuring security and reliability, (2) by reducing costs and waste through battery reuse.

The estimation of the SOH and its prediction (the RUL) is a challenging task. The capacity of a cell cannot be directly measured, so indirect measurements are used instead by using related variables. The SOH can be precisely computed in laboratory conditions, but it significantly differs from the working conditions of real applications [28]. This, unfortunately, does not apply to the real-world EVs that have to employ online estimation algorithms [6]. Battery aging involves many variables, such as charge/discharge current, voltage, and operating temperature. EV batteries' working conditions are also highly dynamic as they change with the environment and the user's driving style [20]. As a result, it is challenging to design accurate physical models due to complex degradation mechanisms and operations. Furthermore, it requires much knowledge about the phenomena involved and experimental data acquired in controlled situations, which could be unavailable or quite expensive to collect [29]. SOH estimation techniques can be classified into two macro-categories: experimental and model-based estimation methods [6], [28]. Experimental methods analyze aging behavior through numerous laboratory tests. As mentioned above, this is typically not achievable on board due to the required equipment and the dynamic driving context. Model-based methods can be further divided into adaptive algorithms and data-driven approaches, and the latter also includes RUL prediction. Adaptive algorithms use mathematical models and numerical filters (e.g., equivalent circuit model and Kalman filters). In contrast, data-driven methods use black box models, which find the mapping between the input and the target. Figure B.1 summarizes the main categories of SOH estimation methods. The following section will focus on machine-learning-based SOH estimation and RUL prediction techniques and their advantages. For a detailed review of the other methods for SOH estimation, please refer to [28], and for RUL prediction, please refer to [103].

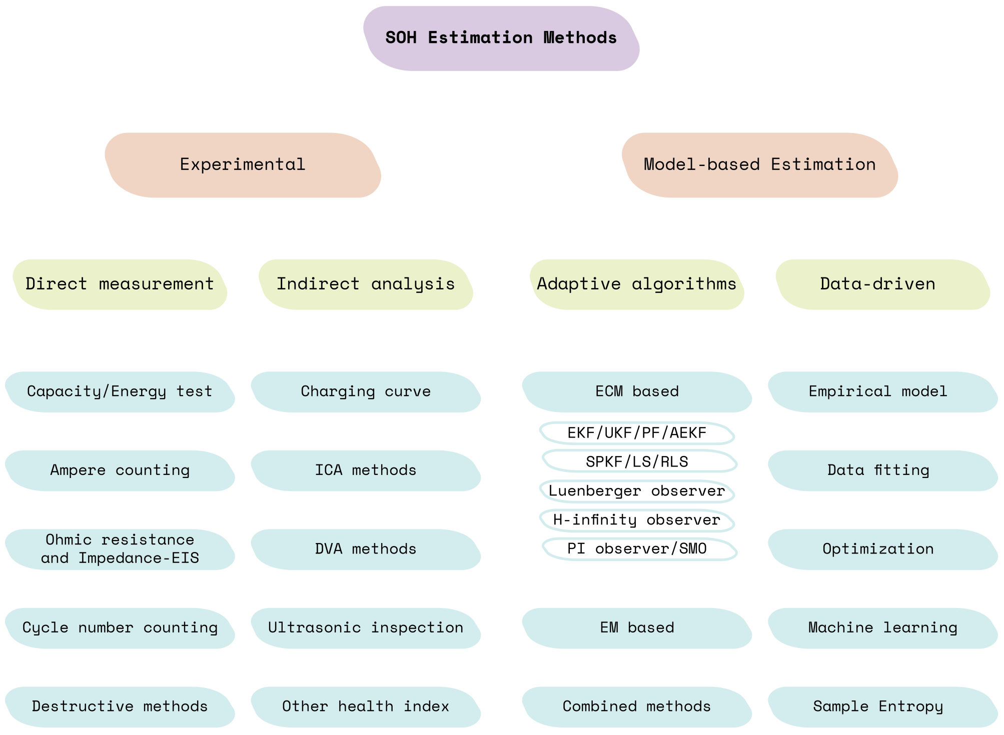


Figure B.1: Classification of battery SOH estimation methods.

This paper contributes in several directions to the applicability of deep-learning-based RUL estimation in practical applications.

- We propose a novel RUL definition in the machine learning context, based on ampere-hours, to push forward the applicability to real cases. The first application of deep learning techniques on an RUL that is not based on the simplified concept of cycles is also provided.
- Two models for RUL estimation are presented and compared on the NASA Randomized dataset: autoencoder plus CNN and autoencoder plus LSTM. In addition, an LSTM is proposed to predict the RUL in the UNIBO Powertools dataset. The results show that the particular autoencoder can effectively extract the relevant features of the cycle curves, while the CNNs and the LSTMs can be used to estimate the RUL.
- Two vast datasets containing batteries cycled with an extensive set of different conditions and variables are used in the experiments to ensure generalization. Compared to the data used in the literature so far (with a limited amount of batteries typically discharged under constant current), the examples used in this paper present many more batteries and conditions that are more challenging to predict. All the relevant details about the data selection and splitting are detailed, ensuring transparency in the results.

The paper is structured as follows. The next section provides an overview of the deep-learning-based techniques used for RUL estimation and the problems that affect their deployment. The relevant works are also summarized. In Section B.3, the datasets and the models used are described. Section B.4 shows and compares the results obtained using the models. Finally, Section B.5 concludes the paper with a summary of the results and future research directions.

B.2 Related Works

The recent success of machine learning in several domains and the availability of data and computing power have motivated the development of novel methods for battery state estimation. Data-driven battery state estimation methods are becoming increasingly popular [13]. In particular, the attention to using deep learning (DL) for battery status estimation has increased over time. Data-driven methods provide several advantages [6]. They allow us to obtain better results in real complex applications, as complete knowledge about degradation mechanisms is still lacking. They do not require expert knowledge about the degradation phenomena as they only rely on enough operational data from which key features are extracted. They are also suitable for execution on hardware with limited capabilities compared to adaptive algorithms that are more computationally demandin [99], [104]. While the training phase is demanding as well, the execution is efficient and can run on BMS hardware, with inference models in the order of a few hundred megabytes [105]. Last but not least, data-driven methods enable the prediction of the SOH (i.e., the RUL), while other techniques are typically limited to estimating the current SOH.

Drawbacks are present, but the benefits compensate for them. The main drawbacks are limited interpretability and inaccessibility to physical parameters (e.g., internal resistance) [35]. Before proceeding, it is worth noting that we may have a conflict of terms. While, in the context of deep learning, "prediction" typically indicates the result of a neural network, in the context of signal processing, it means predicting the future value of a time series. In this paper, we will use the term *estimation* to indicate the estimation of the current SOH. In contrast, we will use the word *prediction* to indicate the expected RUL, as it can be conceived as the prediction of the future SOH.

In the last two years, numerous works using DL for SOH and RUL estimation have been proposed in the literature. In the following, the common approaches (and issues) in the various papers are first presented to avoid repetition, and then the single articles are analyzed.

The variables measured by the BMS are usually voltage (V), current (I), and temperature (T). Such variables are sampled at high rates during subsequent charge and discharge cycles, resulting in very long time series. The variables most used as input are voltage, current, and temperature, as they can capture the battery aging factors [99], but sometimes the sampling time and the state of charge (SOC; i.e., the charge level) are also used. In the case of the RUL, the SOH itself has also been used as input. It is recalled, however, that care must be taken when non-measured variables (i.e., SOC, SOH) are employed, as errors could accumulate and robust estimation of the input variable might not always be available—this is the first issue affecting applicability. The variables can originate from the charging cycles, the discharging cycles, or both. The resulting time series are often presented to the network through a moving window, i.e., the NN makes the estimation based on the set of features at time t plus their last N values. A popular approach is to use a recurrent neural network (RNN)—in one of its various forms—to find the relation between the SOH or RUL and the time series. Long short-term memory (LSTM) [26] networks are particular RNNs that are able to handle long-term sequences and have become the baseline of recurrent networks. LSTMs and their variants are thus also widely used in the battery context. Some experiments try instead to use convolutional neural networks (CNNs) to process time series, or to use a simple feed-forward NN (FFNN) preceded by specific pre-processing. The literature offers several battery datasets to conduct such experiments. One of the most used datasets is the NASA "Battery Data Set" [34]. It was the first battery dataset that was publicly available, and thus it has had a significant impact in the field [33]. The dataset consists of 34 Li-ion 18,650 cells cycled at various ambient temperatures;

however, it includes only constant current (CC)-cycled batteries. Even though the dataset contains 34 batteries, the most common approach is to use a specific subset of three or four batteries. In 2014, NASA released another dataset ("Randomized Battery Usage Data Set") [106] also containing batteries cycled with a random current. A review of battery datasets is available in [33]. The second issue to be addressed to ensure the suitability of the SOH and RUL methods to *real scenarios* is the quality of the dataset used during testing. Most works use *simplified databases* with batteries cycled under CC discharge, a condition not applicable to EV operation. Another applicability obstacle in the case of RUL is in the *definition of RUL* itself. As already mentioned, RUL is defined as the remaining cycles before EOL. In the EV context, we have partial charge and discharge cycles (e.g., discharge to 40%, charge to 80%, discharge to 30%, and so on) as the vehicle can be recharged starting from different SOCs and can be unplugged before the full charge. An equivalent full cycle (0% to 100%) has little practical meaning, therefore the definition of RUL has to be rethought. A valid candidate to represent the RUL in the EV setting is the remaining ampere-hour (Ah) value that the battery can deliver before reaching EOL. The measures to prevent applicability pitfalls are then (1) avoiding using non-measurable variables as input, (2) employing appropriate datasets with wide variability and different conditions, and (3) not using cycles to define the RUL.

The estimation of the SOH has recently become quite robust as it has been applied to realistic datasets. SOH estimation is well established on simplified datasets [6], [13], [107]; here, we report the recent advances on complex datasets. In [92], a hybrid network comprising a gated recurrent unit (GRU; a well-known variation of LSTM) and a CNN is used for SOH estimation. The inputs are the raw V, I, and T data of the charging curve, converted to a fixed size history of 256. The input converges toward the two parallel streams (GRU and CNN) concatenated in the last layer. A maximum estimation error of 4.3% on the Randomized NASA dataset is reported. The authors of [99] proposed an SOH estimation method based on the Independently RNN (IndRNN) and tested it on the Randomized NASA dataset. Here, a discharge cycle is represented by 18 features, including average V, I, and T, as well as the capacity, the time elapsed, and the time periods of each current load. While it achieves superior results, whether it will work in real applications needs to be clarified as it also takes capacity as input. In the experiments, the capacity in input is calculated. In contrast, the proper way to conduct the investigation should have used the capacity estimated by the network itself in the previous time step. In [29], a CNN takes as input the V, I, and capacity of the charging cycle discretized in 25 segments. The output is the capacity computed on the corresponding discharge cycle. Both capacities are computed by coulomb counting. Applicability is at least doubtful in this case too. In [20], a private database of real-world data collected from 700 vehicles (full-electric or hybrid) is used to train an FFNN for SOH estimation. The parameters employed are the accumulated mileage of cars, the C-rate distribution, the SOC range (the SOC is divided into five ranges, and the SOC range indicates such range), and cell temperatures. The number of variables is reduced to a lower dimensionality using principal component analysis (PCA). The results are impressive, with a maximum error of 4.5% and RMSE of 1.1%, which become 2.2% and 0.45% if considering only fully electric vehicles.

Moving to RUL estimation, it has not yet reached the robustness and applicability of SOH estimation. The experiments described in the literature are limited to oversimplified datasets that present only CC-cycled batteries. While adequate performances are typically achieved, there are still some critical issues regarding data quality and applicability. In [36], a temporal convolutional network (TCN) produces RUL estimations. The input is the history of the SOH, processed through a moving window. Tests are performed on three CC batteries from the NASA dataset and two CC batteries from the CALCE dataset. As it uses the history of SOH, a robust SOH estimation is necessary to ensure its applicability. As the experiments rely on ground truth SOH, it needs to be clarified if the proposed method will have the same performance using estimated SOH levels that are thus affected by some error. In addition, a long warm-up is required, as the first output is produced after a minimum t of 30 cycles. Both [37] and [38] propose using an LSTM network that takes the SOH's history as input. In the first case, the output is the RUL; in the second case, it is the k-step ahead SOH, which can be reduced to the RUL. The datasets used are one Panasonic CC battery and four CC batteries from the NASA set, respectively. Both works present the same issues as the TCN-based one (here, the warm-up is even longer). The most promising article is [39], which presents a variant of LSTM, namely AST-LSTM. Two AST-LSTMs are trained, one for estimating the SOH and one for the RUL. The input of the first model includes V, I, T, and the sampling time of the discharge cycles. The second model uses instead the history of SOH estimated from the previous one. As the SOH input is estimated, the RUL approach is also suitable for real scenarios. Experiments are conducted on 12 batteries from the NASA dataset. The approach still needs to be tested on better datasets, and the warm-up is too long. In [84], the IC discharge curve is computed from the V, I, and sampling time. The features extracted from the curve are inputted to a small NN that estimates the SOH and RUL. This method is, however, applicable only to CC discharging. In addition, it has a high computation complexity and low performance. In [40], an autoencoder is used to perform dimensionality reduction starting from the V, I, T, and sampling time of both charge and discharge cycles, plus the capacity estimated during discharge. In addition to the applicability doubts, the accuracy is so low that the approach is substantially inapplicable. Another autoencoder approach from the same main author was proposed in [41]. Here, the autoencoder is used instead to augment the data dimension, and then the result is processed by two branches: an LSTM and a CNN. The features extracted by them are concatenated and fed to the final NN that returns the RUL prediction. While the performance has improved, the dataset used is still insufficient, as only CC-discharged batteries are considered. The properties of the above-mentioned works are summarized in Table B.1.

Method	Input	Output	Dataset	Performances	Issues
TCN [36]	History of SOH	RUL (and SOH monitoring: not considered here)	NASA (3 CC batteries - #5, #6, #18) CALCE (2 CC batteries - #CS_34, #CS_35)	RMSE up to 0.048	• It requires a robust SOH estimation • It is not clear whether it will work if the SOH is affected by errors • Insufficient dataset variability • Long warm-up (minimum starting cycle for NASA is 30 and the starting cycle for CALCE is 360)
LSTM network [37]	History of SOH	RUL	Panasonic 18650 (1 CC battery)	-	• It requires a robust SOH estimation • It is not clear whether it will work if the SOH is affected by errors • Insufficient dataset variability • Long warm-up (start after 50% of battery life)
LSTM network [38]	History of SOH	K-step ahead SOH	NASA (4 CC batteries - #5, #6, #7, #18)	MAE 1.92 (on battery #5)	• It requires a robust SOH estimation • It is not clear whether it will work if the SOH is affected by errors • Insufficient dataset variability • Long warm-up (start at cycle 60)
Variant of LSTM. SOH is estimated and then used as input to predict RUL [39]	V, I, T, sampling time (discharge)	SOH and RUL	NASA (12 batteries)	SOH: RMSE up to 0.059 RUL: RMSE up to 0.026 (on battery #5)	• Insufficient dataset variability • Long warm-up (start at cycle 50)
Features are extracted from the IC discharge curve and given in input to two small NN [84]	V, I, sampling time (discharge)	SOH and RUL	NASA (4 CC batteries - #5, #6, #7, #18)	SOH: MRER up to 1.25% RUL: RMSE up to 5.41	• Applicable only to CC discharging • Computational complexity • Insufficient dataset variability • Low performances
Feature extraction with Autoencoder followed by a DNN [40]	V, I, T, sampling time (charge + discharge), capacity i.e. SOH (discharge)	RUL	NASA (3 CC batteries - #5, #6, #7)	RMSE up to 13.2% (on battery #7)	• It requires a robust SOH estimation • It is not clear whether it will work if the SOH is affected by errors • Insufficient dataset variability • Insufficient performances
Auto-CNN-LSTM: data augmentation with Autoencoder followed by CNN+LSTM extractor [41]	V, I, T, sampling time (charge + discharge)	RUL	NASA (6 CC batteries - #5, #6, #7 #25, #27, #28)	RMSE up to 5.03% (on battery #7, #28)	• Insufficient dataset variability

Table B.1: Summary of current DL-based RUL estimation approaches

B.3 Experiments

In this work, we propose and compare three models to predict the remaining useful life of Li-ion batteries. The contributions and improvements focus on the model's applicability to real-world scenarios and the transparency in defining batteries and methodologies used. Existing works have used limited datasets without specifying why only some batteries from the selected dataset have been used. They also conform to the definition of RUL based on cycles, which has little meaning in actual applications. Finally, most of them do not provide enough information about the data structure and how data are employed. To achieve the desired aims, three aspects have been covered: input definition, output (RUL) definition, and the use of datasets with wide variability.

- **Input:** The only information used is voltage (V), current (I), and temperature (T), as using only measurable variables boosts applicability. The data are taken from the *discharge* cycles and are organized in time series, in the format $[cycle, timestep, variable]$. As explained in Section B.3.2, at each cycle n, the input given to the network is based solely on the cycle n and the previous ones ($n-1, \ldots, n-N$; where N is the history length), i.e., no information from the future is used.
- **Output—RUL definition:** As detailed in Section B.2, defining the RUL as the number of remaining cycles has no practical meaning. Here, instead, the RUL is defined as the *normalized remaining ampere-hour (Ah)* that the battery can deliver before reaching EOL.

This can be named ah-RUL and is computed as:

$$ahRUL(n) = [trapz(current_{[:,:]}, time_{[:,:]}) - trapz(current_{[n:EOL,:]}, time_{[n:EOL,:]})]/nomcap \quad \text{(B.2)}$$

where n is the current cycle number, $trapz$ is the approximate integral via the trapezoidal method, $current$ is the matrix of currents where the x is the cycle and y is the timestep, $time$ is the matrix of the elapsed time corresponding to the current measurement, and $nomcap$ is the nominal capacity of the battery.

As the RUL is a slowly changing value, predicting at every cycle (rather than at each timestep) is more than sufficient. Thus, for each cycle n (and history N), the model predicts the ah-RUL at the current cycle.

The code for the pre-processing, as well as the networks, the trained models, the results, and the plots, are available in the repository of the project, which can be accessed at `https://github.com/MichaelBosello/battery-rul-estimation` (accessed on 15 March 2023).

B.3.1 Datasets

Two datasets have been used. The NASA Randomized Battery Usage dataset [106], which is used to investigate the performances of batteries cycled with a random current, and the UNIBO Powertools Dataset, which was released by our team in [1], and contains batteries with different specifications that were cycled under different conditions.

NASA Randomized Battery Usage Dataset

The NASA Randomized dataset consists of data from 28 batteries. The batteries are lithium cobalt oxide 18,650 cells with a nominal capacity of 2.2 Ah. The cells were continuously operated by repeatedly charging and discharging them according to the corresponding profile. At every 50 cycles, a series of reference charging and discharging cycles were performed to provide battery health status. The batteries are split into 7 groups containing 4 cells each according to the charge/discharge profile and temperature used. The randomized charge/discharge profiles can be as follows: random walk (RW), i.e., the selection of the current is random with a uniform distribution between the two current ranges; skewed RW, i.e., the current selection is random with a custom probability distribution skewed towards lower or higher currents.

- **RW1, RW2, RW7, RW8** batteries are repeatedly charged for a random duration between 0.5 and 3 h, then discharged to 3.2 V using a randomized sequence of currents between 0.5 A and 4 A. The discharge random profile is the RW. The setpoint is loaded every 5 min. Operated at room temperature.
- **RW3, RW4, RW5, RW6** batteries are repeatedly charged to 4.2 V and then discharged to 3.2 V using a randomized sequence of currents between 0.5 A and 4 A. The discharge random profile is the RW. The setpoint is loaded every 5 min. Operated at room temperature.
- **RW9, RW10, RW11, RW12** batteries are repeatedly charged and then discharged using a randomized sequence of currents between -4.5 A and 4.5 A. The charge and discharge random profile is the RW. The setpoint is loaded every 5 min. Operated at room temperature.
- **RW13, RW14, RW15, RW16** batteries are repeatedly charged to 4.2 V and then discharged to 3.2 V using a randomized sequence of currents between 0.5 A and 5 A. The random profile is the skewed high RW. The setpoint is loaded every 1 min. Operated at room temperature.
- **RW17, RW18, RW19, RW20** batteries are repeatedly charged to 4.2 V and then discharged to 3.2 V using a randomized sequence of currents between 0.5 A and 5 A. The random profile is the skewed low RW. The setpoint is loaded every 1 min. Operated at room temperature.
- **RW21, RW22, RW23, RW24** batteries are repeatedly charged to 4.2 V and then discharged to 3.2 V using a randomized sequence of currents between 0.5 A and 5 A. The random profile is the skewed low RW. The setpoint is loaded every 1 min. Operated at 40 C temperature.
- **RW25, RW26, RW27, RW28** batteries are repeatedly charged to 4.2 V and then discharged to 3.2 V using a randomized sequence of currents between 0.5 A and 5 A. The random profile is the skewed high RW. The setpoint is loaded every 1 min. Operated at 40 C temperature.

UNIBO Powertools Dataset

The UNIBO Powertools dataset contains data from 27 batteries featuring various types of cells and experimental conditions collected in a laboratory test by an Italian equipment producer. Cells were charged at 1.8 A until 4.2 V and discharged at 5 A until V_{eod}. Capacity and resistance reference cycles were performed every 100 cycles. The batteries are split into 7 groups, according to the battery manufacturer, the cell type, the cell capacity, and the type of test. The battery manufacturer is defined by a label, either D or E, to keep the brand name private. The cell type defines its intended use, and it can be M (mid-power), E (e-bike), or P (power tool). The tests performed on the cells are Standard (5 A discharge), High Current (8 A discharge), and Pre-conditioned (90 days' storage at 45 degrees C before testing). It follows the list of the groups, with the convention name `XW-Y.Y-AABB-T`, where X is the manufacturer, W is the cell type, Y.Y is the capacity, AABB is the delivery date (AA: week, BB: year), and T is the test type, followed by the list of cell numbers included in the group.

- **DM-3.0-4019-S** 4 cells: 000, 001, 002, 003.
- **DM-3.0-4019-H** 3 cells: 009, 010, 011.
- **DM-3.0-4019-P** 7 cells: 013, 014, 015, 016, 017, 047, 049.
- **EE-2.85-0820-S** 4 cells: 006, 007, 008, 042.
- **EE-2.85-0820-H** 2 cells: 043, 044.
- **DP-2.00-1320-S** 8 cells: 018, 019, 036, 037, 038, 039, 050, 051 (039 has date 2420).
- **DM-4.00-2320-S** 2 cells: 040, 041.

B.3.2 Models

NASA Randomized: AE-LSTM vs. AE-CNN

For the NASA Randomized dataset, we propose and compare two networks. Both models use an autoencoder to compress the long time series in the set, in which timesteps range from 10k to 100k measurements per cycle. Using an autoencoder to perform the reduction allows us to retain most of the information, avoiding the loss of the fundamental information. The prediction of the ah-RUL is then made by passing the sequence of reduced cycles to a subsequent network, a CNN in one case and an LSTM in the other. CNNs are known to perform better on structured data with local information, which motivates their use in this use case. We also considered LSTMs as they are designed to handle long-term sequences such as the battery aging process. Thus, they can learn the long-term degradation trend of batteries.

As mentioned above, the autoencoder has to reduce the thousands of values per cycle to a small number of features (in the order of dozens) that are representative of the cycle. This is done by an NN with an hourglass shape trained to copy its input to its output (in this case, the cycle measurements). As there is a bottleneck in the middle layer, it will learn a compact representation of the input that retains most information. To obtain the best reconstruction results, we employed a specific autoencoder structure presented in [108] that retains both *local and global information*. The autoencoder structure is depicted in Figure B.2. This network uses skip connections, i.e., jumps, to keep the features extracted not only from the last layer but also from the middle one.

The encoder takes the time series of one cycle as input, composed of 11,800 measurements with 3 values each: voltage, current, and temperature. Voltage, current, and temperature are normalized between 0 and 1 using min-max normalization to prevent the magnitude of the value from affecting its importance. The min and max values are computed from the training set only. Depending on the experiment, the length of the time series in the group could be in the order of 10 k, 20 k, or 100 k. To reduce all the series to the same size, time series with 20 k measurements were sampled, keeping 1 value for each 2, and similarly, 1 value for each 10 was kept for the 100 k time series. After that, the exceeding length was cut to 11,800. The series goes into a 1D CNN with 16 filters, a kernel size of 10, and 5 strides. It follows max pooling with size 5. Here, the skip connection opens a fork: the features extracted so far are given to both a CNN layer and a flattening layer followed by a dense layer. This dense layer with dimension 7 gives the first part of the encoded vector containing the local information. Coming back to the other path, the CNN has 8 filters with kernel size 4 and 2 strides, and is followed by max pooling with dimension 4, a flattening, and a dense layer with size 7 that produces the last part of the feature vector. The decoder performs the inverse operations mentioned above without the skip link. All the layers use the ReLu activation function. Adam is used to train the network with a learning rate of 0.0002, MSE loss, 500 epochs, and batch size 32.

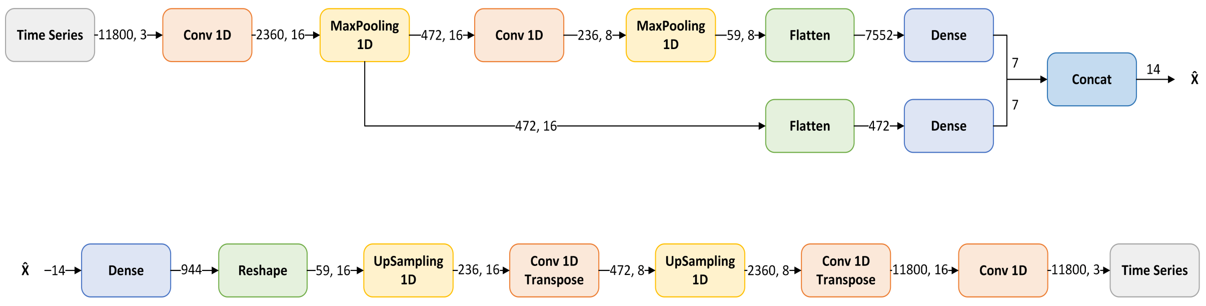


Figure B.2: The structure of the autoencoder used to compress the cycles. The skip link allows us to retain both local and global information

Once the autoencoder has been trained, the encoder is used to reduce the cycles to a feature vector of size 14. The feature vector is again normalized between 0 and 1, and then the time series for the RUL prediction is formed. The subsequent networks take a time series of $N = 1000$ in the case of the CNN and $N = 500$ in the case of the LSTM. The time series is formed by concatenating the vectors of the current cycle n plus the previous N cycles. If the current cycle number n is below N, the time series is padded with zeros. A warmup of 15 cycles in training, and 30 in testing, is given to the network, i.e., the predictions start at cycles 15 and 30, respectively. The output of the nets, detailed at the beginning of the section, is also normalized between 0 and 1.

The CNN used has two 1D convolutional layers, with 64 and 32 filters, respectively, kernel sizes of 8 and 4, and 4 and 2 strides. It follows the flattening and three dense layers, with dimensions of 32, 16, and 1. All the layers use L2 kernel regularizers and ReLu activation, except for the last one, which uses linear activation. The Adam optimizer is used with a learning rate of 0.000003, Huber loss, 3000 epochs, and a batch size of 32.

The LSTM network has one masking layer to ignore padding zeros, two LSTM layers with 128 and 64 neurons each, and three dense layers with 64, 32, and 1 neuron each. All the layers again use L2 kernel regularizers but SELU activation, and linear activation in the last one. The same Adam optimizer is used with a learning rate of 0.000003 and Huber loss, but 500 epochs are performed.

UNIBO Powertools: LSTM

In the case of the UNIBO dataset, the autoencoder is not used since the cycle already has a low dimensionality. The length of the cycles of this dataset is reasonably short, in the order of 300 measures per cycle. In this case, using an autoencoder does not provide significant benefits in terms of dimensionality reduction, while it would add unnecessary complexity to the data processing pipeline. Simpler preprocessing has been employed instead. In particular, the discharge cycle is reduced to six features: the average V, I, T, and their standard deviations. The LSTM network used is the same as presented for the NASA Randomized dataset, considering both structure and hyperparameters. Likewise, the same history length, warmups, and normalization are employed.

B.4 Results

B.4.1 NASA Randomized

Batteries from the different groups (as detailed in the previous section) have been used in both training and testing, to ensure reliable results. Six out of seven groups have been used. The group of batteries that have been cycled using the RW on both charge and discharge has not been used because it produced too much data to be handled (having a lot of micro-cycles). Plus, it can be considered an unrealistic use. Therefore, batteries RW9, RW10, RW11, and RW12 have not been used. Of the remaining 6 groups, each having 4 batteries, 3 batteries have been used for training and 1 battery for testing, with the following exceptions. The battery RW3 has been excluded because the temperature measurement is corrupted, and the battery RW20 has been excluded because all the measurements are 0 for almost all of the battery life. As a result, the groups of the batteries RW3 and RW20 had only 2 batteries used in training, making the learning even harder for such groups. The total number of batteries was 16 in training and 6 in testing. The complete list of the batteries used in training and testing is available in the project repository. The exact same splitting of batteries is used in the training and testing of both the autoencoder and the ah-RUL predictor, as it is assumed that at application time, neither of the networks will know the new data.

To evaluate the performance of the autoencoder and the ah-RUL predictor, the average root-mean-squared error (RMSE) is employed, as defined by the following formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{B.3}$$

The average is computed on the RMSE obtained on each cycle of each battery. The reconstruction error of the autoencoder achieved a notable 0.0356 average RMSE on the testing set. The Autoencoder is thus able to retain the most important information of the curves at every phase of the battery life and cycled under different conditions, and then reconstruct them with high accuracy. An example of the original and reconstructed voltage, current, and temperature curves of a battery in its middle life is shown in Figure B.3.

The CNN-based and the LSTM-based ah-RUL predictors achieved comparable results. The CNN obtained an RMSE of 0.0799, while the LSTM attained a 0.074 RMSE. Figure B.4 compares the testing results of the CNN and the LSTM, taking as an example two batteries from different groups: a battery from the skewed-high RW at room temperature and a battery from the skewed-low RW at a temperature of 40 degrees C. While the LSTM achieved slightly better results, the CNN provided more stable and consistent results over time. This is evident by the plots showing that the CNN prediction curves are always well-fitted, even though they could have an offset compared to the real curve. The LSTM, instead, does not suffer from the offset but has much more irregular curves. The plots for all the batteries are published in the project repository. It is worth remarking that the remaining ampere-hour is predicted instead of the number of cycles. Considering that, and considering the very diverse conditions applied to the batteries, the results of both networks are excellent, although with different strengths, i.e., well-fitted curves with offsets versus no offset but with irregular curves.

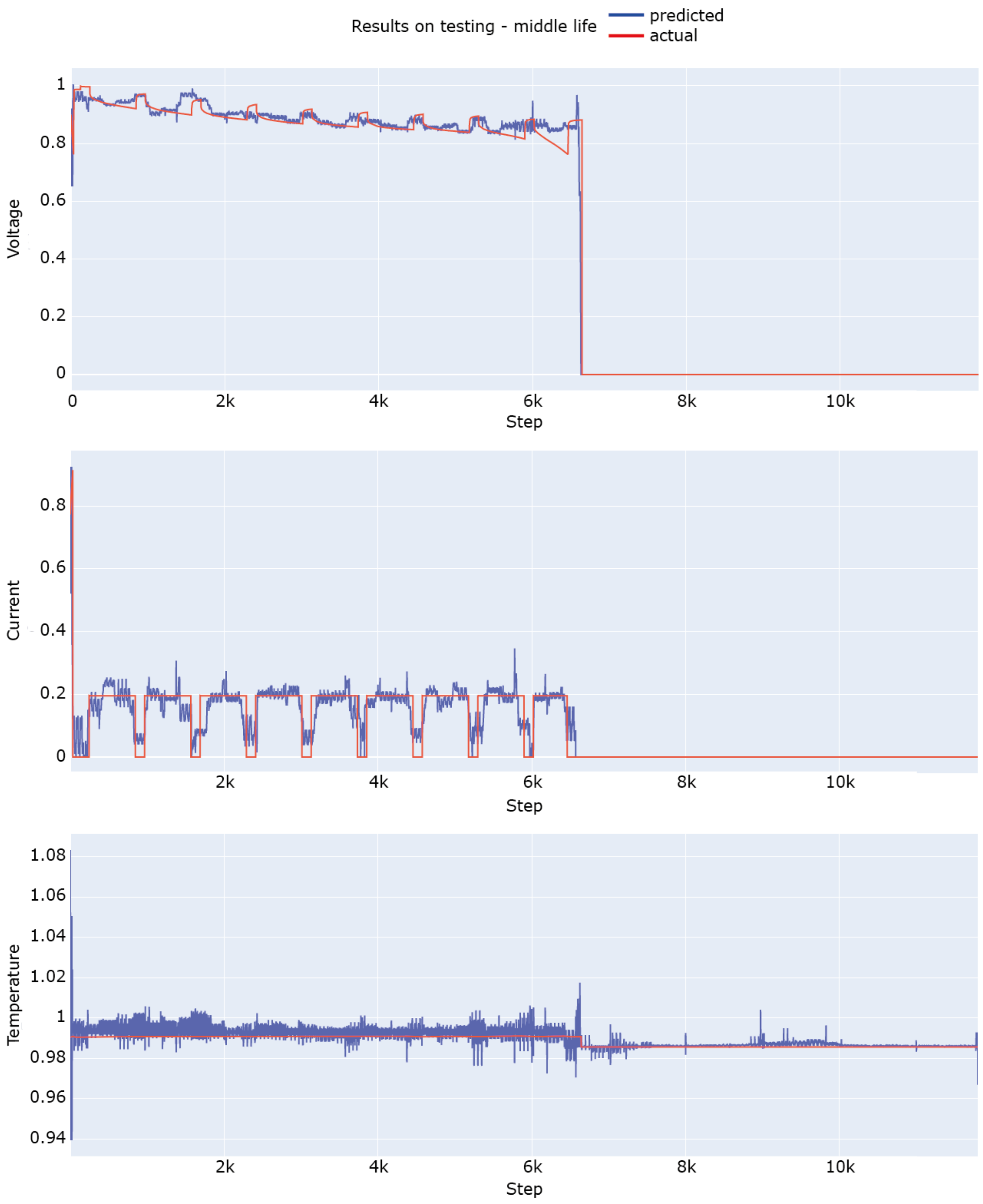


Figure B.3: Example of the results of the autoencoder reconstruction capabilities on the testing set. The voltage, current, and temperature curves of a cycle in the battery's middle life were reconstructed from the extracted features.

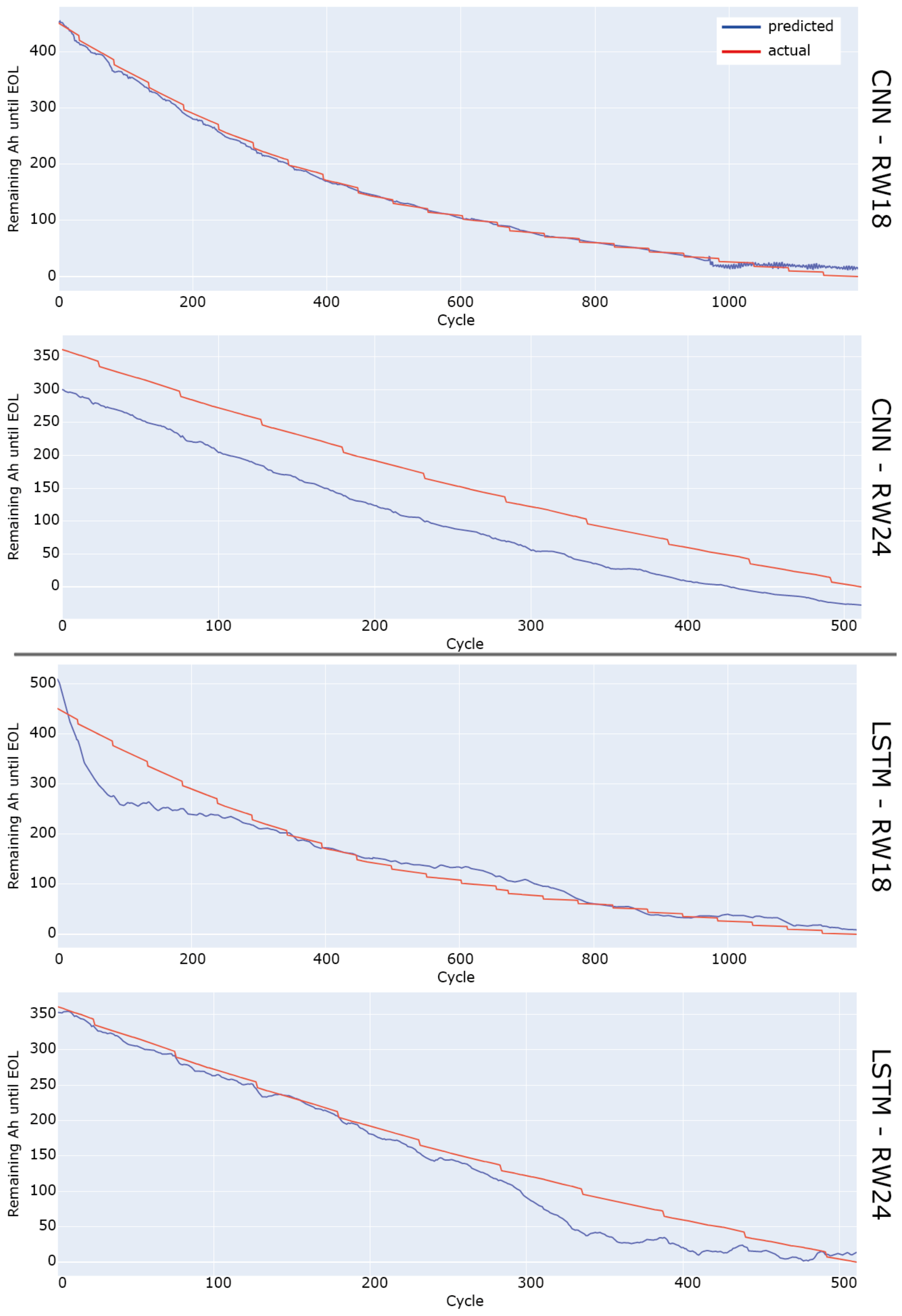


Figure B.4: A side-by-side example of the results of the CNN and the LSTM in ah-RUL prediction on the NASA Randomized dataset. The two examples show better stability and curve fitting of the CNN but a lower offset from the LSTM.

B.4.2 UNIBO Powertools

In the UNIBO experiment, the batteries in the training and testing sets also came from all the group types to guarantee the fairness of the results. One battery from each of the 7 groups was selected for the testing set. All the other batteries were put in the training set. Batteries 047 and 049 were excluded as, at the time of the dataset construction, they were not cycled yet until end-of-life. Battery 019 was not used as some of its data are corrupted. This resulted in 7 batteries for the testing set and 20 for the training set. The LSTM achieved an average RMSE of 0.021. An example of the prediction on the testing set is shown in Figure B.5. The results demonstrate the ability of the LSTM to learn the degradation trends of the batteries cycled under different conditions.

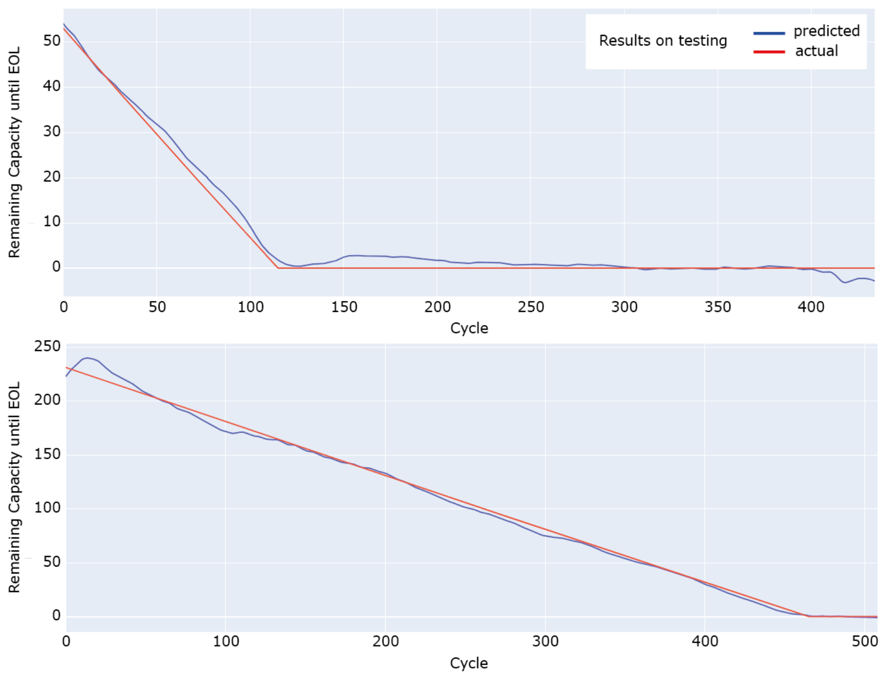


Figure B.5: An example of test results of the LSTM ah-RUL prediction on the UNIBO Powertools dataset.

B.5 Conclusions and Future Works

In this paper, we proposed and compared two models using the NASA Randomized Battery Usage dataset for the RUL estimation of Li-ion batteries based on an autoencoder plus CNN and an autoencoder plus LSTM. In addition, an LSTM was proposed to estimate the RUL in the UNIBO Powertools dataset.

Aiming to push forward the applicability to real cases of the current deep-learning-based methods for RUL estimation, we proposed a novel definition of RUL based on ampere-hours, which is more useful for real scenarios. We also employed two datasets with a wide range of cycling conditions to ensure the generalization of the methods. Compared to the datasets used in the literature so far, which employ a limited amount of batteries typically discharged under constant current (a condition that is not realistic in EVs), the NASA Randomized dataset provides an entirely different difficulty in predicting the RUL. The UNIBO Powertools dataset also provides a fresh perspective on data diversity as it contains batteries from various manufacturers and with different specifications.

To the best of the authors' knowledge, this is the first successful application of deep-learning-based methods for RUL estimation on such a vast number of Li-ion batteries cycled under different conditions, and the first study to use the UNIBO dataset. This is also the first application of those methods on an RUL that is not based on the simplified concept of cycles. The results show that the particular autoencoder employed can extract the dominant features of the cycle curves, and that both the CNN and the LSTM proposed can predict the RUL based on those features.

Several directions are open for future investigation. While the results obtained on such complex data are encouraging, there is still room for improvement. It would be interesting to design a network that provides the advantages of both the CNN (well-fitted curves) and the LSTM (no offset). In addition, transformers [109], which have achieved impressive results in recent times, should be studied. This kind of NN can learn temporal dependencies that are even longer than the ones learnable by an LSTM. Therefore, it is quite promising for the objective. To the best of the authors' knowledge, this kind of NN has not been tested yet for RUL estimation. The transformer will substitute the current LSTM, in the hope of achieving even better performances.

The improvement of battery RUL estimation can support the development of battery recycling. Still, policies will be required to fully develop it, either with economic incentives [110] or with the involvement of governments [111]. Further work in this direction is also advised.

Author Contributions: Conceptualization, M.B. and C.F.; methodology, M.B.; software, M.B.; validation, M.B.; formal analysis, M.B.; investigation, M.B.; resources, G.P.; data curation, M.B.; writing—original draft preparation, M.B.; writing—review and editing, M.B.; visualization, M.B.; supervision, G.P. and C.R.; project administration, G.P.; funding acquisition, G.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the LiBER project under the POR FESR program by Emilia Romagna Region, years 2019–2022

Data Availability Statement: Data are available from the sources cited in the article. The UNIBO Powertools dataset is available at https://doi.org/10.17632/n6xg5fzsbv.1 and the Nasa Randomized Battery Usage dataset is available at https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository (accessed on 15 March 2023).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix C

Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1^{tenth} LIDAR-Based Races

Reprinted, with permission, from:

M. Bosello, R. Tse, and G. Pau, “Train in austria, race in montecarlo: Generalized rl for cross-track f1tenth lidar-based races,” in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2022, pp. 290–298. DOI: 10.1109/CCNC49033.2022.9700730

Train in Austria, Race in Montecarlo: Generalized RL for Cross-Track F1^{tenth} LIDAR-Based Races

Michael Bosello, Rita Tse, Giovanni Pau

Abstract – Autonomous vehicles have received great attention in the last years, promising to impact a market worth billions. Nevertheless, the dream of fully autonomous cars has been delayed with current self-driving systems relying on complex processes coupled with supervised learning techniques. The deep reinforcement learning approach gives us newer possibilities to solve complex control tasks like the ones required by autonomous vehicles. It let the agent learn by interacting with the environment and from its mistakes. Unfortunately, RL is mainly applied in simulated environments, and transferring learning from simulations to the real world is a hard problem. In this paper, we use LIDAR data as input of a Deep Q-Network on a realistic 1/10 scale car prototype capable of performing training in real-time. The robot-driver learns how to run in race tracks by exploiting the experience gained through a mechanism of rewards that allow the agent to learn without human supervision. We provide a comparison of neural networks to find the best one for LIDAR data processing, two approaches to address the sim2real problem, and a detail of the performances of DQN in time-lap tasks for racing robots.

C.1 Introduction

Autonomous driving (AD) has remained a principal societal goal envisioned since the automotive infancy [112]. Despite recent successes in partial automation [113], [114], AD is still a quite complex problem for both traditional control algorithms and Machine Learning (ML) techniques [45]. In contrast, driving is spontaneous and semi-automatic for humans. Understanding and reacting to stimuli is a natural behavior for us that is learned during infancy and refined through *experience*. Developing an artificial agent able to fully replicate this behavior is one grand challenge of this century.

The first and most prominent impact of ML on AD has been in recognition and scene understanding. Convolutional Neural Networks (CNNs) [115] had been a breakthrough in pattern recognition [105] affecting several fields and overtaking typical Computer Vision approaches that were not enough robust for AD. The superior robustness and better performance of Deep Learning (DL) combined with large datasets [116] lead to the wide use of ML in the field. DL models could require a vast dataset and even several weeks for training, however, the resulting model is relatively small, typically in the order of a few hundred Megabytes [105]. Also, the prediction phase has relatively low requirements for the hardware, opening to real-time applications. Perception tasks like line and vehicle detection have been made possible by CNNs which allow performing such tasks at frame rates suitable for AD [44].

Supervised learning is the most common ML approach to Autonomous Vehicles (AVs), widely supported by automotive companies. Over 40 companies are working on Automated Driving Systems, including traditional automotive players and novel ones (e.g. Audi, Tesla, Waymo) as well as tech companies (e.g. Apple, Amazon, Nvidia) [117]. The approaches adopted in the industry vary in terms of sensors and control techniques. The vehicles can be instrumented with a wide range of sensors including RGB and depth cameras, LIDARs, GPS/GNSS receivers, IMUs, and they can have multiple sensors of the same kind in different positions and with different features e.g. Field of View, resolution, etc [118]. On the software side, the typical strategy is to use ML models for perception (and sometimes for prediction) to interpret the scene while using hand-engineered policies for control. Those ML models, trained using supervised learning, demand enormous amounts of labeled data. The training phase becomes very expensive especially when labeling can not be automated thus requiring considerable human effort. AD systems that rely on high definition maps crafted using dense sensors are also getting momentum in industry [119]. Maintaining such complex maps and the supervised ML models lead to a burden on the communication and computational infrastructure. Sensors costs, operational costs, as well as data collection, transmission, and processing costs, are driven higher.

A different vision advocates for the use of Reinforcement Learning (RL), where an agent learns through experience by interacting with the environment in a *trial-and-error* manner. It thus does not require explicit human supervision. While supervised learning techniques learn input-output associations (i.e. they are not able to learn the dynamics of an agent's environment), RL is specifically formulated to handle the agent-environment interaction [46]. It is therefore a natural approach for learning robotics (and autonomous driving) control policies. In the context of robotics applications, RL has taken an increasingly important role as it allows us to design hard-to-engineer behaviors [120] and to optimize problems that have no closed-form solutions. The interest in training RL agents directly in real-world robotic systems (rather than training in a simulator and transferring the learning) is also growing [121] as transferring the experience from a synthesized environment to a real scene has proven itself a difficult task [42]. The employment of RL in the real world is spreading also to AD with the first application of RL in a real car for line following [47]. In the same line, we want to explore the use of RL for AVs in the real world but using a small-scale yet realistic model to keep the costs reasonable and ensure maximum safety. Despite the small scale, the driver agent has to address all the challenges and uncertainties deriving from the noises and the unpredictability of sensors and actuators.

In our previous work [48], we have explored the use of RL directly in the real world using a small-size robot-car instrumented with a wide-angle camera, and we have shown that Deep Q-Network (DQN) can be trained in a real-time system using real-world camera frames. In this work, we have built a very realistic 1/10 scale racing car (namely, an f1tenth [122], [123]), equipped with a LIDAR, and capable of performing DQN training in real-time. We have then trained the agent to drive on both real and simulated race tracks using LIDAR data as input. This paper presents three contributions:

- Compares the performance of one-dimensional convolutional neural networks (1D CNN), two-dimensional convolutional neural networks (2D CNN), and fully-connected networks when used to detect LIDAR data.
- Introduces two approaches to face the simulation to real-world (sim2real) problem: *(i)* training the agent directly in the real world using *real* LIDAR data as input, which has been considered an *open problem* because of LIDAR's faults and noises [51]. *(ii)* training the agent in the simulator and using the model in F1*tenth* car without any retraining thanks to our LIDAR pre-processing algorithm.
- Demonstrates the race-performance, sample efficiency, and generalization capability of DQN through simulations in challenging F1 racetracks.

The rest of this paper is structured as follows: subsection C.2.1 gives a background on RL-based autonomous racing while in C.2.2 we examine the challenges of using LIDAR data in the context of RL, and the work done on this subject. In section C.3, we briefly introduce RL and define the driving problem in the RL setting. The setup used in our experiments is described in C.4 while in section C.5 we discuss the results obtained. Finally, section C.6 closes the paper with the final remarks and future works.

C.2 Related Works

C.2.1 RL for autonomous racing

The great success of the DQN algorithm [124] proposed by Deep Mind, which reached super-human level control on Atari games, has forested the use of RL also in the autonomous driving domain. RL has been used in both end-to-end learning (where the model maps the scene straight to the proper action) and in single-task handling (where it is applied to one or more AD sub-task, typically in the control part). For example, multiple works applied RL to racing games using both methods [125]–[127]. It is worth noting that video games are an easier environment (having discrete-time events and simplified mechanics) compared to proper simulators featuring a continuous control setting. RL has been used also to solve single tasks like motion planning, overtaking, intersections/merging, lane change, lane keep, and automated parking [42].

While supervised learning can be used in real cars, RL has been usually applied to *virtual environments* because of its trial-and-error nature that leads to safety concerns and considerable expenses. however, simulated and real data have not the same distribution, and agents trained in a synthesized world often *fail to generalize to the real scene* [42] if appropriate domain adaptation measures are not taken. Both low-dimensional sensors like cameras and high-dimensional sensors like LIDARs are affected by the issue [51], [128], thus, several methods to address the sim2real problem and transfer the learning from simulation to the real world have been proposed. As mentioned in the introduction, researchers have started to explore the training of RL agents directly in the real world. This is a different vision to how the sim2real problem is usually faced. Authors of [47] used a continuous action RL algorithm, namely Deep Deterministic Policy Gradients (DDPG), to learn a policy for lane following. The input comes from a front-facing camera and the rewards are proportional to the distance traveled without human intervention. The training was carried out in a real car using on-board computation and the model was able to learn to follow the lane in a road segment in just half an hour.

C.2.2 LIDAR-based RL

LIDAR [129], [130] (an acronym for light detection and ranging) has become an established method for measuring distances by emitting focused beams of light and measuring the time it takes for the reflections to be detected by the sensor. The measured distances can be dense, resulting in high-resolution maps. LIDAR sensors can be either 2D or 3D. 2D LIDARs produce a map of the azimuth at a fixed height while 3D LIDARs produce 3D point clouds. In our case study, we employ a 2D LIDAR. An example of LIDAR data sensed by our car is shown in section C.4.

Thanks to the progress in the LIDAR technology and the use of DL, LIDAR has become widely used in autonomous driving [131], but not in the context of RL-based driving. While the use of camera frames in RL driving tasks has become rather common, works using LIDAR data to guide RL driving agents have been proposed only recently in the literature.

Authors of [51] used RL to train several NNs having LIDAR data as input in the context of NN verification, taking the f1tenth platform as a testbed. The system has been trained and verified in a simulated environment, after that, the sim2real gap has been analyzed. The environment consisted of a single 90-degree right turn, with the car approaching and passing the turn at constant throttle. When they moved to the real environment, they found that LIDAR measurements are greatly affected by reflective surfaces (a problem that we faced as well). When a ray gets reflected, it appears as it is no obstacle in that direction. The agent performed well in the ideal simulated environment. In the real environment with all the reflective surfaces covered, LIDAR faults caused crashes in 10% of the runs. In the unmodified real environment, they were not able to train a robust controller using state-of-the-art RL algorithms. They have therefore claimed it has been an open problem.

Brunnbauer et al. [132] presents one of the best studies up to date on the use of RL on the F1tenth platform. The task is to drive as fast as possible without collisions on racetracks using LIDAR data as input. An agent is trained on an F1tenth simulator to run on complex F1 racetracks (scaled to 1/10th) using a model-based RL algorithm (Dreamer). The race performances, sample efficiency, and generalization abilities of Dreamer have been assessed by showing how it outperforms the five baseline model-free RL algorithms tested by the authors. The sim2real transferability was also evaluated by demonstrating that trained Dreamer can be deployed to the real F1tenth in a physical test track. This work will be used as a competitor of our model. In our experiments, we will show that DQN (which is model-free) can achieve better race performance and sample efficiency than Dreamer while keeping the same generalization capabilities and sim2real transferability.

In [133], the sim2real transferability of a LIDAR-based driving agent (as well as image-based and segmentation image-based agents) is analyzed. The LIDAR-based agent trained on a realistic simulator using DDPG was not able to run directly in a physical test track. To bridge the sim2real gap it has been necessary to tune the driving parameters and use a sort of hack, i.e., scaling down the LIDAR measurements to prevent crashes. [134] proposes two stabilization approaches to smooth the learning of a driving RL agent. The simulated car equipped with a LIDAR has been trained in a simple track (harder tracks have been tested in a video game that should not be considered). In [135] LIDAR measurements are plotted to an image that is used as input to DQN. The car was able to navigate in a simulated track and a simple L-shape physical maze.

C.3 Reinforcement Learning Primer

In this section, we first provide a brief overview of RL [46]. Second, we analyze how to model the driving problem in an RL context.

RL is a machine learning paradigm in which an agent learns how to accomplish a task by interacting with its environment. The agent-environment interaction is defined by three signals; at each time step the agent performs an *action* and the environment responds with a *state* and a *reward*. The state is a set of information about the environment the agent can observe and use to predict the future. The rewards define how good the behavior of the agent is towards solving its task and it is thus the way the developer can specify the task to be achieved. The agent uses the states as a guide to select the most appropriate actions aiming to maximize the *cumulative reward*, i.e the sum of rewards *over time*. As the actions performed by the agent affect the evolution of the environment, it has to balance between actions with high immediate rewards and actions that lead to favorable states with significant future rewards. The behavior of the agent is produced by a *policy* which is a function that associates states to actions. The policy that maximizes the cumulative reward is called optimal policy and it is the one we want the agent to learn.

One of the method to create a policy is to *estimate* the *action-value function*. This function associates state and action pairs to a value called *expected return*. The expected return is an estimation of the cumulative reward i.e. it is the sum of rewards the agent expects to get starting from a state S and following a policy π. Once the agent has estimated the action-value function, it only needs to select the action with the greatest expected return. *Q-learning* is an algorithm to estimate the action-value function of the optimal policy. At each iteration, the algorithm refines its estimation of the function using the new experience produced by the agent-environment interaction. The more precise the estimation is, the more the policy gets closer to the optimal one. Q-learning uses a table to represent the action-value function but this is not practical when the state space grows. This problem can be solved using a NN instead of a table. The NN will *approximate* the Q-function and possibly converge to the real function. This is not just useful for the dimension problem but it opens also to generalization as the experience gained in one state can be used for similar ones. Using NNs instead of tables in q-learning is known as Deep Q-Network.

In the training phase, the agent has to explore the environment by performing different actions to improve the estimation of the value function. Balancing exploration and exploitation is a known problem of RL. In this work, we use one of the typical strategies to face this problem, the ϵ-greedy policy. An agent that follows an ϵ-greedy policy has a probability $1 - \epsilon$ to behave greedily and a probability ϵ to choose a random action.

C.3.1 Modeling the driving problem

A Markov Decision Process (MDP) completely formalizes an RL problem. An MDP is a tuple composed of the set of states, actions, immediate rewards for each state transition $s \to s'$, and the state transition probability function which specifies the probability that performing an action a in the state s will lead to the state s'. The fundamental property of a state, known as the Markov property, is that it can be used to predict the future. A stochastic process (i.e., an environment) has the Markov property if the conditional probability distribution of the future state s_{t+1} depends only upon the current state s_t and action a_t, not on the sequence of events that preceded it. In many cases of interest such as autonomous driving, the agent has only *partial* information about the world and it receives only some observations about the environment instead of the complete state. In such cases, the agent tries to reconstruct the state obtaining an approximation of it that may not be Markov. In some situations, the Markov property could be partially dropped, but performance may dramatically worsen if no appropriate measures are taken. In a Partially Observable MDP, the Markov property is assumed to hold in the underneath process, but the agent is not able to observe all the variables of the environment. To reconstruct the state, the agent integrates the information over time e.g. by keeping a history of observations or by using a Recurrent NN.

Properly characterizing the driving task requires close attention as developers have a great deal of freedom in choosing input features (states) and control signals (actions) [42]. Moreover, the driving problem is open to subjective definitions as different behaviors in similar situations may be equally acceptable [116]. As a result, designing a suitable reward function could be tricky. The correct design of the reward function is crucial to obtain the desired behavior, especially in real-world systems [42]. A set-up of driving as an MDP is given in [42], [47], [136].The definition of the states, actions, and rewards used in our experiments are detailed in the next section.

C.4 System Overview

C.4.1 The F1tenth platform

We used the F1/10 (or f1tenth) platform [122] as a testbed for our experiments. F1tenth is an open-source 1/10 scale race car designed for autonomous systems and cyber-physical systems research. This platform offers high-performance and hardware/software stacks similar to full-scale solutions while limiting costs and dangers typical of real vehicles. To conduct real-world experiments, the f1tenth features *realistic dynamics* like Ackermann steering and the ability to achieve high speeds. Remarkably, authors of the platform have shown that it is possible to port the stack on a real car. Ackermann steering is the geometric arrangement of linkages used in the steering of a typical car. Compared to differential steering, which allows a vehicle to turn by applying different drive torque to its sides, Ackermann steering has a more complex dynamic model and requires a wider operating space. In our previous experiment [48] we used differential steering which required less engineering efforts to design the learning environment and that resulted in less training time. For example, a vehicle with differential steering can recover from bad states near an obstacle by turning on itself, whereas a car with Ackermann steering will eventually reach a fatal state. This kind of dynamics makes the agent learn slower and, if care is not taken, the agent might not learn at all. The f1tenth platform is much more realistic than typical robots employed in navigation which deliver slow speeds and differential steering.

The bottom chassis of the f1tenth comes from a 1/10 scale race car available from Traxxas. The Traxxas model has very realistic mechanisms, just like a real race car. It has a brush-less motor capable of reaching (depending on the model) over 90 km/h, and a servo motor for steering control. The top chassis is a custom laser-cut ABS plate that hosts all the electronic components. The power board is a PCB designed to provide stable voltage to the car and its peripherals, and it is connected to a Lithium Polymer battery. The VESC (Vedder Electronic Speed Controller) controls the speed of the motor and the direction of the servo. The mainboard that controls the car is a Jetson TX2 [137], a GPU module designed to enable embedded AI. Its GPU and memory capabilities make it possible to train NNs in real-time. The TX2 module is housed on a carrier board [138] characterized by a reduced form factor. Both the hardware and software stacks of the f1tenth are modular. Several sensors are available on the f1tenth, but only the LIDAR is present in our build. Fig. C.1 shows our produced f1tenth. The 2D LIDAR used is a Hokuyo UST-10LX [139]. The sensor has a 270° field of view with an angular resolution of 0.25°, for a total of 1081 scan rays. It has a detection range of 10m, an accuracy of ±40mm, and a scan speed of 25ms. According to the datasheet, a specific number is returned if a measurement error occurs like there is no object in the detection range or the object has low reflectivity.

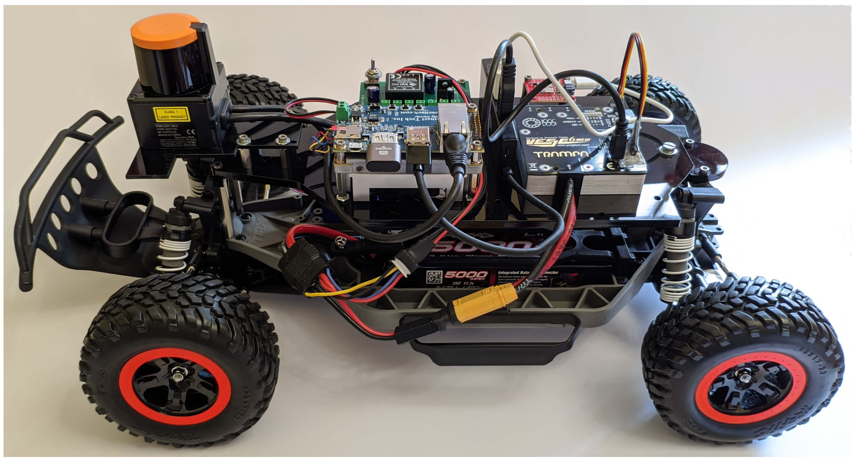


Figure C.1: The f1tenth race car used in the experiments.

On the software side, sensors, actuators, and controllers are represented as nodes and coordinated by ROS (Robot Operating System) [140], a middleware strongly based on the publish/subscribe paradigm. Even though ROS provides several facilities, only hardware abstraction and communication have been used in our context. A very handful feature of the f1tenth is the possibility to run your code on both the real f1tenth and the provided realistic simulator, almost without changes. The simulated hardware implements sensors and actuators nodes with the same interfaces as the real car. The simulated environment is rendered using rviz, a visualization tool for ROS. The biggest difference is the field of view of the simulated LIDAR which is 360°. We reduced it to 270° like the Hokuyo to conduct experiments with the same conditions.

C.4.2 RL environment

As mentioned above, the software is coordinated through ROS. The RL environment implements the controller node that communicates with the – simulated – hardware nodes. The node uses rospy, the Python controller for ROS. In rospy, the callbacks associated with a subscriber are executed in a separate thread, while a publisher sends messages from the main thread. Publishers and subscribers are Python objects that a node can create. The RL environment is built as follows. Two subscribers receive respectively LIDAR and odometry data. The RL environment uses the latest stored data, which are updated asynchronously by the subscribers at every new measurement. A publisher, in a separate thread, sends the latest driving command chosen by the RL agent to the VESC. The publisher sends the command at fixed time intervals and continues to send it until the agent asynchronously updates the command. A safety callback implements the automatic emergency braking and ensures that the car will not – hardly – crash. Every time it receives a LIDAR scan message, the time-to-collision is calculated on every ray. If the time-to-collision is below a threshold, the car will brake with a higher priority compared to commands sent by the RL agent. The time-to-collision is calculated using the following equation: $TTC_i(t) = r_i(t)/[-v_x cos(\theta_i)]_+$ where r_i is the distance measured by ray i, v_x is the car's linear velocity (obtained from the odometer), and θ_i is the beam angle. At each cycle, the RL environment executes the provided action and returns a state, a reward, and a boolean that indicates if the episode ended. If the emergency braking was activated, the episode ends and a negative reward is given. Before starting a new episode and returning the control to the agent, the car goes in reverse to ensure it has enough space to start again. The **state** is formed by the two latest raw LIDAR scans (plus the car's velocity in the third experiment setting). The agent has three **actions**: forward, right, and left (plus a slowdown action in the third experiment setting). Having few actions let the agent learn faster. Considering that the commands update is quite fast, there is no need for more actions as the agent can still achieve appropriate control. We tested several **reward** functions during our experiments. We started with a simple reward function in the first experiment and ended with a more appropriate function for the second and third ones. In both instances, the car gets a negative reward (-1) when the emergency braking is activated, and rewards are clipped between $[-1, 1]$ because it could dramatically affect the learning efficiency [124]. In the first reward function, a fixed reward is assigned to each action: the forward movement gives a reward of 0.2, while turn and left give a 0.05 reward. This differentiation prevents zigzagging behaviors which are a form of reward hacking [141] i.e. the agent finds a way to get more rewards but with a detrimental behavior (in this case by running in a zigzag pattern). In the case of the second reward function, the reward is proportional to the car velocity obtained through odometry, with a maximum of 0.09. This *optimizes* for the fastest route. A small bonus, with a maximum of 0.01, proportional to the distance to the nearest obstacle is added to the previous reward to award safer routes.

C.4.3 LIDAR data pre-processing

The Hokuyo LIDAR has 1081 rays with an angular resolution of 0.25°. Due to the curse of dimensionality, each additional value in the input vector makes the state space grow exponentially. To reduce the state size, we have cut the field of view from 270° to 180° as we are not interested in overtaken obstacles. In addition, we have further reduced the state size by grouping values together. In our experiments, the f1tenth's hardware was able to manage in real-time a NN that can process at most 40 rays, and the best results were obtained with 20 rays. Moreover, a coarse-grained resolution is sufficient for racetracks. We used the exceeding rays to strengthen the measurements by grouping the rays together. We have tested three grouping method: `average`, values are split in 20 groups and averaged; `sampling`, a value each x is taken; `minimum`, the minimum of the group is taken. As the average method seems to be the most robust, it is the method used in the experiments. Values are also scaled between 0 and 1 (min-max normalization), with 1 corresponding to the maximum measurable distance, as in backpropagation having input values in the range $[0, 1]$ improves learning [142].

C.4.4 From simulation to reality

The greatest challenge when moving to the real car is given by LIDAR faults and noises. This is true in both the proposed sim2real approaches i.e, directly training the agent in the physical world or transferring the learning from the simulator. As explained in section C.2.2, LIDAR faults could be tedious for correct control. Since we already achieved resilience to noise with the pre-processing, at this stage we needed only to filter out the bad readings (the ones with the error value specified by the manufacturer) from the groups. Thanks to the grouping method, removing bad readings does not affect the resulting vector. This was sufficient to train/transfer a robust controller.

When it comes to training the agent in the physical world, two more issues should be considered: the cycle timing and the reset mechanism[121]. In the *continuous time* settings, the clock is ticking independently of the agent behavior. As a result, the agent cycle should have the right timing or it will not learn a robust policy. If the cycle is too fast, the agent will not be able to see the changes in states and to relate the consequences of its actions. If the cycle is too slow, the agent will not be able to correlate states as changes would be too substantial. The cycle speed is a problem also for the reward signal. The faster the cycle, the more the reward received; vice-versa, the slower the cycle, the fewer the rewards received. Obtaining an appropriate timing required to optimize the code to speed up the cycle in the car as it has limited hardware as well as slowing down the cycle in the simulator by adding a sleep. Regarding the reset mechanism, when an episode ends because of emergency braking activation, the car goes backward for a fixed time before starting a new episode. This mechanism was chosen because it is feasible in both simulation and real-world. When more complex scenarios are approached, a new reset mechanism will have to be conceived instead.

C.4.5 Driver-agent software

The algorithm used is DQN [143] along with standard enhancements that make it effective like experience replay, target network, and state history. The main parts of the agent's software are the NN and the training cycle.

The NN approximates the Q-function by taking a state as input and giving as output a vector formed by the expected return for every available action. The details of the NNs used are discussed in the next section.

The training cycle starts with the action selection following an ϵ-greedy policy. The agent chooses the action with the greatest expected return inferred by the NN with probability $1 - \epsilon$ and a random one with probability ϵ. The selected action is then performed and the environment responds with a new observation vector, a reward, and a boolean value indicating whether the episode ended. The observations are pre-processed as stated in section C.4.3 and two subsequent observations are stacked together to form the state. This integration over time (state history) allows to detect movements of surrounding objects but also of the car itself. Training is performed using *transition samples* which contains a state with its reward, the performed action, the state reached after that action, and the terminal indicator. The information of the sample allows to compute a revised expected return – improved thanks to the new experience – and train the network on it through gradient descent. A new sample is produced at every iteration step through the new experience and it is added to the *replay buffer*. The buffer stores all the samples obtained from the interaction with the environment. If the maximum capacity of the buffer is reached, the older samples are discarded. The training starts when the replay buffer is filled with a minimum number of samples called observation steps. After the update of the replay buffer, a training step is executed on a random *batch* of transition samples. Using a random batch allows using the samples multiple times capitalizing on them and also to break their temporal relation ensuring that the batch is i.i.d (identically and independently distributed), a fundamental condition for Stochastic Gradient Descent (SGD) effectiveness. This technique is called *experience replay*. As having a moving target makes NNs unstable, two NN are used to reduce the instability: the target network and the behavior network. The target network is the one updated at every cycle through gradient descent while the behavior network is the one used to select the actions and produce behavior. The behavior network is then updated every few steps by copying the target network's weights.

The source code, the trained models, and the Tensorboard logging of experiments are available here[1].

C.5 Results and Discussion

This study about RL analyzes three factors. The first one assesses the best NN for LIDAR data processing. The second one verifies two sim2real approaches. The third one shows the race performances, sample efficiency, and generalization abilities of DQN in complex F1 racetracks. All the experiments use the following parameters: the learning rate is 0.00042; gamma is 0.98; epsilon goes from 1 to 0.1; the other hyperparameters can be consulted in the repository. It should be mentioned that using Huber loss [74] had a greatly positive effect on learning.

C.5.1 NN comparison experiment

The NNs tested have been: 1D CNNs, fully connected networks, and 2D CNNs. A set of NNs having a range of hidden layers up to 6 and units per layer up to 256 have been tested for each category. The NNs reported below are the respective best ones. The comparison experiment has been conducted in simulation to have a consistent and uniform environment between runs as well as to use a complex track. The track is shown in fig. C.2. The task is to run on the track without collisions. The speed is set to 1/3 of the maximum car speed.

[1]https://github.com/MichaelBosello/f1tenth-RL

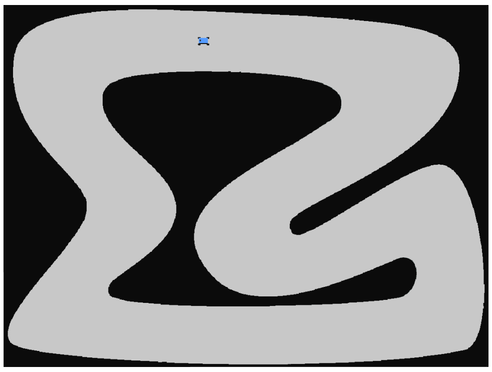


Figure C.2: The simulated racetrack used in the comparison experiment.

The 1D CNN has two convolution layers with respectively 16 filters with kernel size 4 and 32 filters with kernel size 2. It follows a flatten layer and a dense layer with 64 neurons. The fully connected network has two dense layers with 128 units each. The 2D CNN operates on black and white images generated from LIDAR data (grid maps) obtained by outlining the measured borders. The coordinates of the white pixel corresponding to one beam are calculated as follows (where the zoom factor helps to separate borders):

$x = value_i * zoomfactor * cos(angle_i) + offset$

$y = value_i * zoomfactor * sin(angle_i) + offset$

The network is composed of a convolution layer with 16 filters and kernel $(4, 4)$; a max-pooling layer with kernel $(2, 2)$; another convolution layer with 8 filters and kernel $(2, 2)$; another max-pooling layer with kernel $(2, 2)$; a flatten layer; a dense layer with 64 units.

CNNs perform better on structured and *spatially related* data and they are typically used on inputs with local information (i.e. data structured as an array where position matters) such as images, audio, and text [144]. *1D CNNs have proven to be very effective in processing LIDAR data* as they have been the best performer in our tests. The agent equipped with a 1D CNN also showed a behavior oriented to cut curves to minimize turning actions and maximize forward actions.

The plot in fig. C.3 shows the average cumulative reward on 100 episodes of the three NNs during the training stages and the evaluation phases, that alternated train stages. The 1D CNN learned a robust policy in almost 870 episodes with a total training time of three hours, and it reached a maximum of 430 average cumulative rewards in the evaluation stage. The dense network performed well with 1030 training episodes in three hours and a maximum cumulative reward of 230. The 2D CNN learned a decent control policy after 1870 episodes with a total training time of four hours, and a maximum cumulative reward of 98. We have not been able to make the 2D CNN learn a robust policy. Moreover, It should be noted that the 2D CNN is much more hardware demanding, and it could be hardly suitable in embedded systems like the f1tenth.

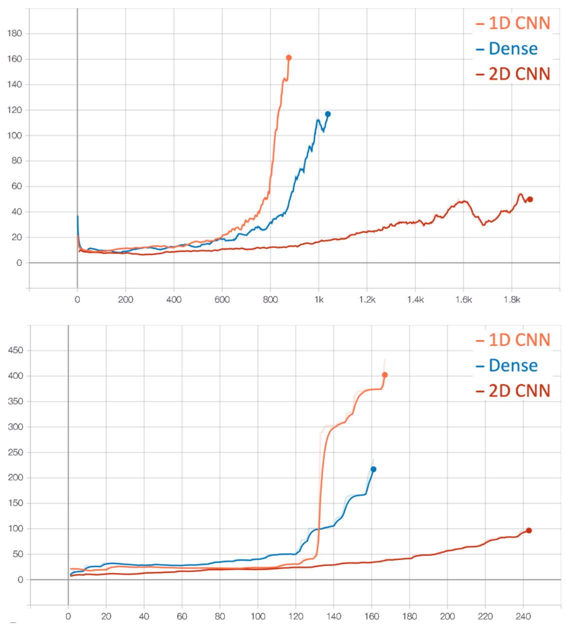


Figure C.3: Comparison of the average cumulative reward on 100 episodes of different NN in the training (top) and evaluation (bottom) phases

C.5.2 Sim2real experiments

Training on the physical car

This experiment aims to verify whether training of DQN could be performed directly in a real environment with noisy LIDAR data. Due to space constraints, we had at the time, the physical track of this experiment is a simple round track. The track along with a visual representation of data produced by LIDAR scans is shown in fig. C.4. The challenges addressed to train a robust controller in the real world have already been discussed in section C.4.4. The model used in the sim2real experiments is the 1D CNN explained above as it is the best option.

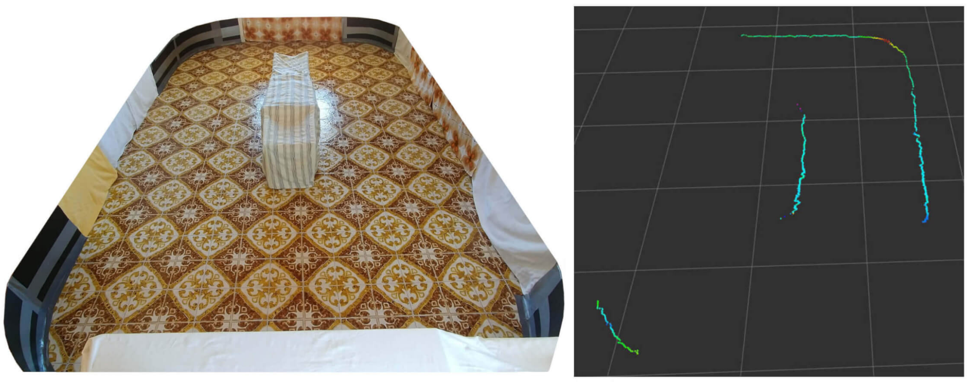


Figure C.4: The track used in the real-world training experiment and the visual representation of the perceived LIDAR data.

The physical f1tenth successfully learned a robust control policy to drive in a simple track in three hours. The training has been done at 1/8 of the maximum car speed. We have verified that the policy can scale to higher velocities without re-training. In this track, the higher velocity reached has been 1/4 of the maximum car speed. A demonstrative video about the evolution of the car's behavior is accessible in the project's repository. The results are evident in the plot of average cumulative reward during training which is shown in fig. C.5 (it should be noted that the rewards obtained in the simulator and the real environment have different magnitudes as the cycle time is different. The average cumulative reward of 40, obtained in the real car, is quite good compared to the one of the random policy which is 5).

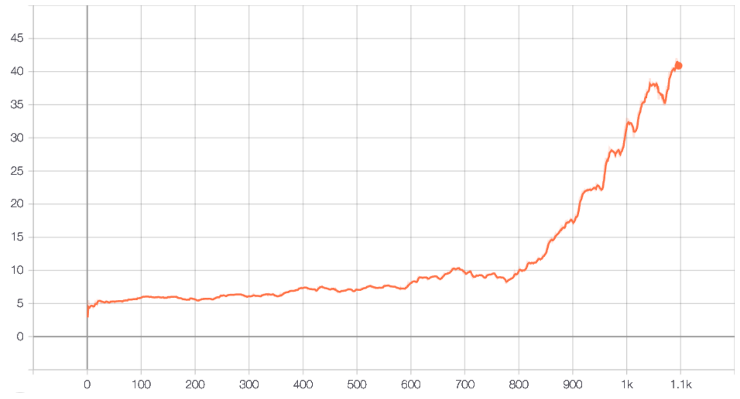


Figure C.5: Average cumulative reward on 100 episodes in the training phase of the real car experiment.

Transfer learning

For the transfer learning experiment, we employed a wide and complex-shaped area, the roof of our university. To keep from building an actual race track, the perimeter is used as the track the car should follow. To let the car follow the perimeter, the emergency braking is activated also when all the LIDAR measures are greater than a threshold. Therefore, when the car goes too far from the perimeter, the episode ends with a negative reward. This configuration has no impact on the overall system and the car can learn how to use the perimeter as a racetrack.

The area has been mapped through the car's LIDAR using the Hector SLAM [145] package available in ROS. The area and the map are shown in figure C.6. The map has been used to train the agent on the simulator and then move the model to the physical f1tenth. Thanks to the realism of the F1tenth simulator and our pre-processing that prevents the disruption caused by noises and faults of the real LIDAR, the trained model was able to control the real-world car *without any additional training or changes*. We tested the F1tenth at a speed up to 1/3 of the maximum for security reasons, although it is likely that the policy could scale to higher velocities. The video of this experiment will be available in the project's repository as well.

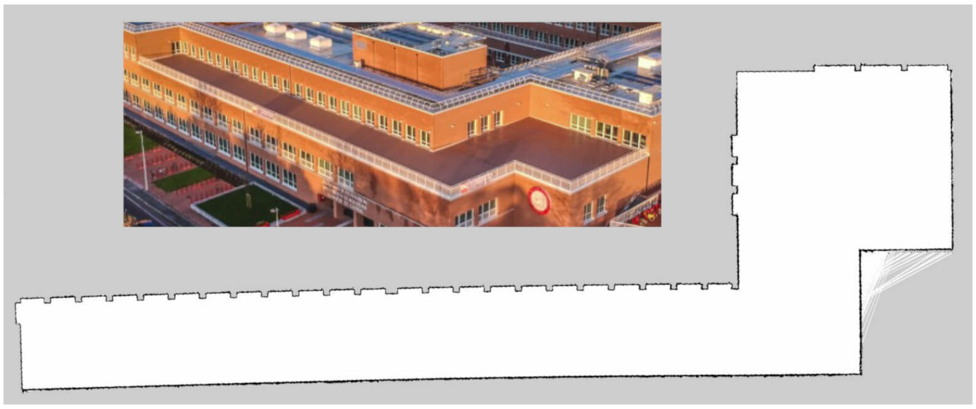


Figure C.6: The roof of our university used as a track and the map resulted from Hector SLAM.

C.5.3 F1 racetracks experiment

The agent task in this experiment is to complete laps as quickly as possible in complex F1 racetracks (scaled to 1/10th) replicated on the F1tenth simulator. The racetrack difficulty is evident by analyzing their minimal track width, track length, and minimum curve radius [132], [146].

To make the agent competitive, the speed limits are removed and two changes are adopted. A slowdown action is added to the basic three – forward, right, left –. This action keeps the last direction but reduces by half the throttle (if it is greater than a minimum threshold). The state is extended with the linear velocity of the car, which is computed using the car odometer, to let the agent know when to use or not the slowdown action. The state is thus composed of the latest two LIDAR measurements and the latest two car's velocities. The NN used is the same as the 1D CNN explained in the first experiment but with the addition of a direct link from the two velocities in input to the last dense layer of the network.

The agent has been tested on four F1 tracks that are displayed in figure C.7. The agent has been trained only on the AUT track and it reached top results on all the four racetracks, demonstrating how *DQN can generalize very well on unseen hard tracks*.

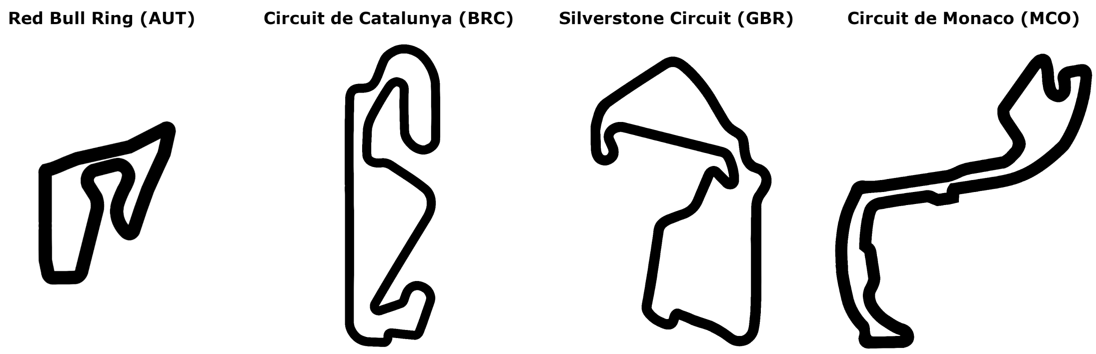


Figure C.7: The F1 tracks used in the experiment.

The *performances (i.e., lap time) and sample efficiency* of DQN are evident from the test results showing that our model outperforms the algorithms presented in [132] in both terms. The authors of [132] evaluated their model-based RL algorithm (Dreamer) and five model-free alternatives (D4PG, PPO, LTSTM-PPO, MPO, SAC) with the same experimental setting used by us where the agents were trained on AUT and then tested on other tracks. Regarding the sample efficiency, Dreamer was trained for over 2 million steps and the model-free agents for more than 8 million while our model required only 550 thousand steps. Moving to the race-performance analysis, our model can achieve better times compared to both Dreamer and the other model-free competitors. Over ten trails, our agent was always able to complete the laps in under 23s in AUT, 56s in BRC, 48s in GBR, and 42s in MCO. Furthermore, Dreamer was not always able to complete the laps while the other model-free algorithms were not able to complete the lap-time tasks at all (see [132]). The comparison of the lap times of our agent and the other algorithms is plotted in figure C.8. The chart shows the results of the algorithms that were able to complete the task: DQN, Dreamer, and Follow The Gap (a baseline algorithm for robot navigation). Only the results on AUT and BRC were available in [132].

While authors of [132] claimed that model-free RL can not complete the tasks when the complexity of the track increases, thus advocating for the better performance of model-based RL, our results demonstrate that model-free RL can compete with and even surpass model-based RL, while keeping the same generalization capabilities and sim2real transferability.

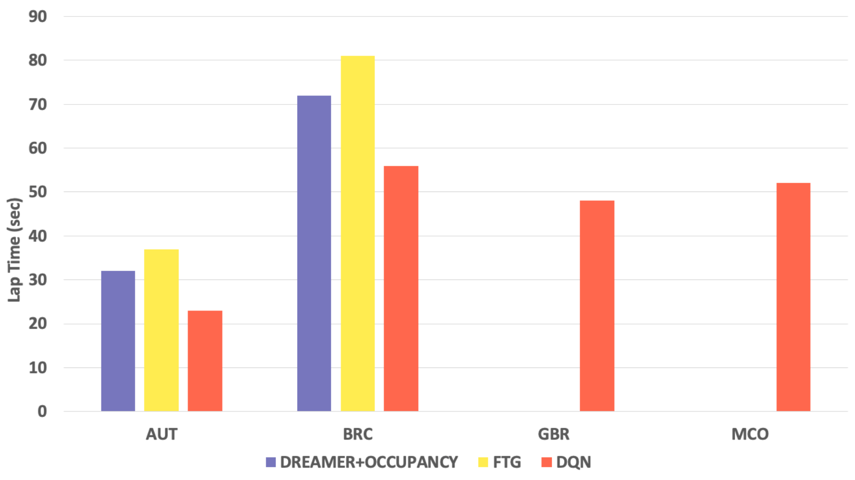


Figure C.8: The lap times achieved by the best algorithms on four tracks (the lower, the better).

C.6 Conclusions and Future works

In this work, we provide three contributions to the study of RL-based racing cars. Our comparison experiment has shown that 1D CNNs are particularly effective in LIDAR data processing. The sim2real experiments have shown that: (1) it is possible to train a robust controller using DQN directly in the real world and in the presence of LIDAR faults, thanks to appropriate pre-processing. A problem previously considered not solved. (2) A model trained in simulation can be transferred to the physical car without additional training and changes when a realistic simulator and a robust pre-processing are used. Finally, the race-performance, sample efficiency, and generalization abilities of DQN are demonstrated in complex simulated F1 scenarios, outperforming the alternatives.

We aim at continuing our investigation in several directions that can help to understand how close to real urban scenarios it is possible to push this approach without sacrificing safety. Specifically, we plan to act in several directions:

- Testing multi-modal approaches using both LIDAR and camera frames.

- Implementing a deep RL technique for continuous action space like Asynchronous Advantage Actor-Critic [147].
- Experimenting with meta-learning [148] approaches and verify if they could be useful in the context of AVs.
- Move to an urban scenario and make the agent follows high-level directions to reach a target.
- Performing *cooperative learning* by exchanging car's experience with efficient communication [149].
- Using Imitation Learning [150] to speed up learning in the initial phase through human demonstration and then refine the policy through RL

Thanks and Acknowledgements

This work was supported in part by the Macao Polytechnic Institute - Edge Sensing and Computing: Enabling Human-centric (Sustainable) Smart Cities (RP/ESCA-01/2020).

This work set the foundation of Bosello's master thesis [136] and is partially reported in it. While in that document the focus is on the integration of cognitive agents and RL, with the F1tenth being only a testbed, here we discuss how RL can be applied to real-world LIDAR data and how the various models and options perform.

Appendix D

Enabling Deep Reinforcement Learning Autonomous Driving by 3D-LiDAR Point Clouds

Reprinted, with permission, from:

Y. Chen, R. Tse, M. Bosello, *et al.*, "Enabling deep reinforcement learning autonomous driving by 3D-LiDAR point clouds," in *Fourteenth International Conference on Digital Image Processing (ICDIP 2022)*, vol. 12342, SPIE, 2022. DOI: 10.1117/12.2644369

Enabling Deep Reinforcement Learning Autonomous Driving by 3D-LiDAR Point Clouds

Yuhan Chen, Rita Tse, Michael Bosello, Davide Aguiari, Su-Kit Tang, and Giovanni Pau

Abstract – Autonomous driving holds the promise of revolutionizing our lives and society. Robot drivers will run errands such as commuting, parking cars, or taking kids to school. It is expected that, by the mid-century, humans will drive only for their pleasure. Autonomous vehicles will increase the efficiency and safety of the transportation system by reducing accidents and increasing the overall system capacity. Current autonomous driving systems are based on supervised learning that relies on massive, labeled data. It takes a lot of time, resources, and manpower to produce such data sets. While this approach is achieving remarkable results, the required effort to produce data becomes a limiting factor for general driving scenarios. This research explores Reinforcement Learning to advance autonomous driving models without labeled data. Reinforcement Learning is a learning paradigm that uses the concept of rewards to autonomously discover, through trial & error, how to solve a task. This work uses the LiDAR sensor as a case study to explore the effectiveness of Reinforcement Learning in interpreting complex data. LiDARs provide a dynamic high time-space definition map of the environment and it could be one of the key sensors for autonomous driving.

D.1 Introduction

Automated driving has a high social actual value [151]. It can reduce the incidence of traffic accidents and effectively alleviate traffic congestion. Many traffic accidents result from erroneous or dangerous driving behaviors, such as fatigue driving and drunk driving. The application of self-driving can greatly reduce these situations and accidents caused by unskilled driving techniques. According to SAE's definition of Autonomous Driving (AD) technology, as the intelligence of cars increases, automatic driving can be divided into five levels,1 starting from no automation (L0), passing through driving assistance, partial automation, conditional automation (L1, L2, L3), and ending in high/full automation (L4, L5). The most cutting-edge technologies are now focused on the last two stages, L4 and L5. With the increasing level of AD, the real-time perception accuracy and sensitivity requirements of the surrounding environment (roads, pedestrians, vehicles, buildings, etc.) are getting higher.

The driver-less technology uses various algorithms to automatically control a moving vehicle to achieve the same driving effect as a driver [152]. As Fig. D.1 shown below,autonomous vehicles drive by repeating three high-level tasks [116]: (i) Recognition; (ii) Prediction; and (iii) Decision Making. Each of those tasks comprises several algorithms and processes like sensing, localization, and communication. The resulting vehicle control entails actual driving (controlling the engine, steering the wheel, using the brake, etc.) to ensure it is always safe, as well as to follow the correct directions. In order to successfully implement these three steps, the following five sub-modules need to be completed. First, in the sense module, the car perceives the surrounding environment through multiple sensors. In the perceive and localize module, the car applies different algorithms to extract information from the sensory data then analyze the environment. The next task is to understand the scene and predict the possible evolution of the environment. After the execution of the above modules, the unmanned vehicle will generate a driving strategy and plan the path. The last module is to control the different behaviors of the car while following the computed path.

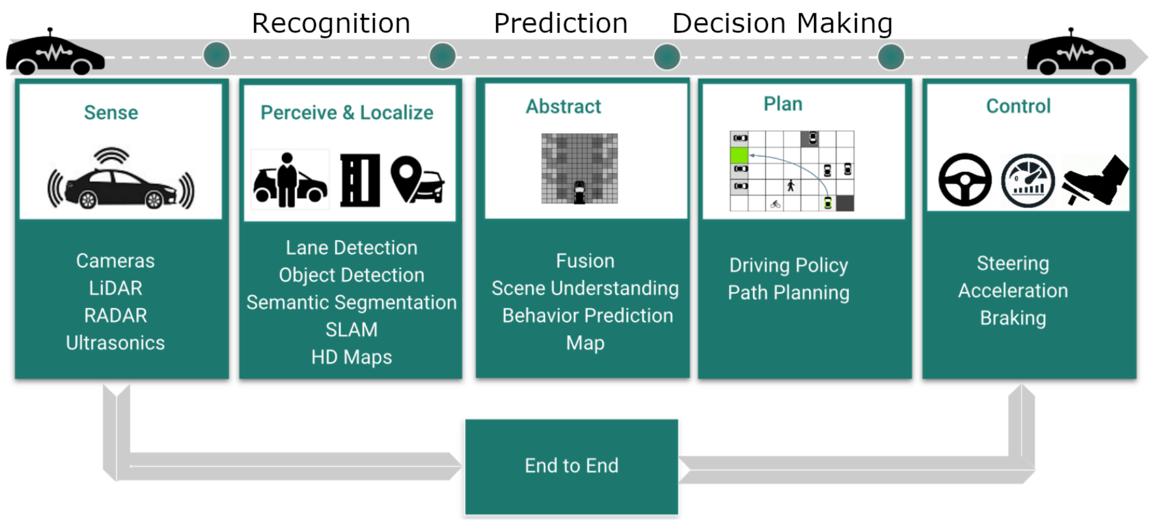


Figure D.1: Modules of autonomous driving systems (Source: Talpaert et al. [116] - modified)

The previous approach used to realize the aforementioned modules is supervised learning, especially for the recognition/tracking and prediction processes [45]. Nonetheless, such a method requires huge, labeled datasets, resulting in a burden on the computational and communication infrastructure. The cost of obtaining labels is also quite expensive. The label typically needs to be added manually, which costs a lot of labor. A technique that does not require labels could help to cover more and more scenarios, thus obtaining better generalization. According to this idea, Reinforcement Learning (RL) is a candidate to advance AD. RL is a learning paradigm in which an agent learns how to fulfill a task by trial & error [46]. Unlike supervised learning, which learns input-output associations [153], a RL agent relies on the experience obtained by interacting with the environment, and its behavior is guided by the reward signal that evaluates the agent's actions.

As one of the core sensors in current AD technology, LiDAR plays a key role in the perception of the surrounding environment [154], thus it can provide appropriate environment representations for our RL agents. The LiDAR [129], [130], an acronym for Light Detection And Ranging, measures the time it takes a light beam emitted by the transmitter to hit a target and come back to the sensor. From the Time-of-Flight (ToF), it is possible to compute the distance from the detected surface. A 3D-LiDAR can emit thousands of high-frequency beams at different vertical and horizontal angles. The result is an accurate 3D structure information of the object or scene [155], expressed as a point cloud. An example of a rendered point cloud of a driving scene is shown in Fig. D.2. The point cloud is made up of a set of three-dimensional points. The information in the cloud includes the location of each point, i.e., the x, y, and z coordinates in the three-dimensional space. In addition, there can be color details, light intensity, category label, and other information [156]. Therefore, the general point cloud shape format is $[N, M]$, where N is the number of points, and M can be akin to the total of channels in the image. The point cloud can depict the characteristics of a sparse 3D world, and it can be used to perceive obstacles through the method of classification and clustering. With the breakthrough in detection and segmentation technology brought by Deep Learning (DL), LiDAR has been able to efficiently detect pedestrians and vehicles, and output detection boxes, namely 3D bounding boxes [157], or output labels for each point in the segmented point cloud [158].

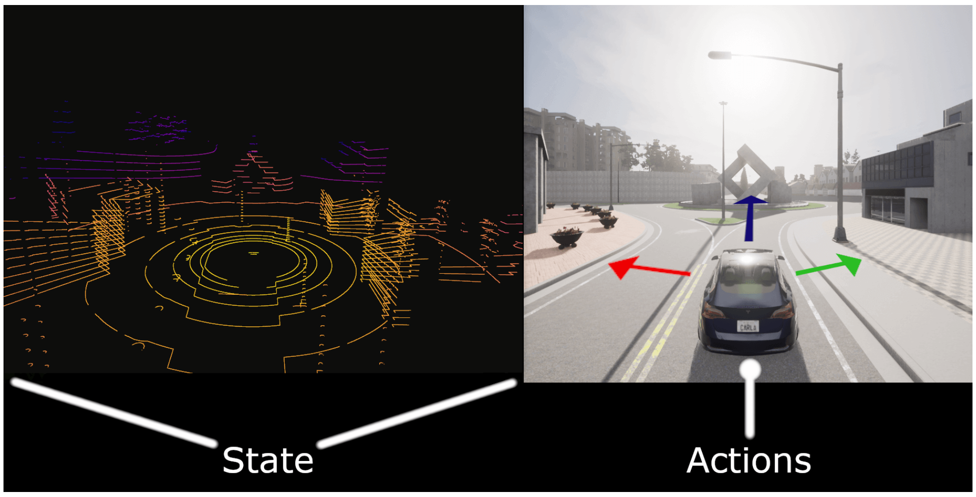


Figure D.2: Lidar view in CARLA with the corresponding scene from RGB camera

LiDAR is a valuable sensor due to its dense and fast measurement capabilities, and high accuracy. Therefore, LiDAR is a promising sensor for highly and fully automated AD, and the research on it could lead to improvements in the field. After obtaining the point cloud data from LiDAR, this research proposes a process to employ point clouds in RL for AD training to exploit the advantages of RL which does not need labels and can learn directly from the experienced situation. Since that, it does not need direct human supervision and it is particularly effective in agent-based tasks learning.

The article evolves as follows. The related work section introduces the literature related to this research. The methodology section illustrates the algorithms and methods employed. The experiment section explains the proposed method and the realized software, as well as the experiments performed to achieve the research goals. The conclusion section sums up the results and puts forward prospects for future research directions.

D.2 Related work

D.2.1 Reinforcement Learning in autonomous driving

Deep Q Network (DQN) [143] has been one of the breakthroughs of Deep Reinforcement Learning. Its core ideas are the following three points: use rewards to construct labels through Q-Learning; use experience pool to solve the correlation and non-static distribution problems; use a Convolutional Neural Network (CNN) – the MainNet – to generate the current Q value and use another CNN – Target – to generate the target Q value. DQN is nowadays a powerful baseline agent. As a first version attempt, DQN has achieved versatility in the game field. Since its success in several complex tasks [143], [159], [160], RL has been also widely studied in AD tasks. Initially, RL has been applied to complete racing games using end-to-end learning [127], [161] (i.e. controlling the car starting from the raw data without splitting the task) or to complete driving sub-tasks like lane-following, merging, overtaking, parking [42]. More recently, more refined approaches were proposed that can transfer the learning of a simulated car to real prototypes [3], [42], [132] or train the RL agent directly in real-world robotic systems (rather than training in a simulator and transferring the learning) [47], [48], [136], [162]. In the following, we present the front-line works on AD and RL whose ideas we consider more promising and profitable.

Waymo published ChaufferNet [162] at the end of 2018, introducing how to use and improve Imitation Learning to obtain a more robust driving model. Different from the typical end-to-end learning, it uses a mid-mid method, which performs well in both the simulation environment and the actual vehicle test. The model not only imitates and learns reasonable driving behaviors based on a large amount of data, but also creates various special driving situations by adding disturbances to the reasonable driving trajectories and combines the corresponding additional loss functions to train the network how to deal with disturbances and avoid bad behaviors.

Xu et al. [163] correlated the robustness of the driving strategy to the changes in the vehicle dynamics model, such as changes in model parameters (mass, moment of inertia, tire model parameters, etc.) and external disturbances such as road inclination and sideways. The authors indicate that the trained driving strategy can be directly applied to vehicles that have certain changes relative to the one in the training environment, such different vehicles can be simulated or real and still achieve the same effect as in the training environment. In order to achieve this goal, the author proposes a driving strategy transfer framework based on robust control (RL-RC). They train the initial RL strategy in the simulated training environment (source domain), and then apply the trained agent to the target domain.

Wang et al. [164] published Vision-Language Navigation in 2019, such as an active learning scenario where linguistic semantics and visual perception are combined to teach an agent how to move in a 3D environment. The authors propose a new Reinforced Cross-modal Matching method, which can promote both local and global cross-modal reference through RL. To improve the generalization of the learned policies, the authors further proposed a Self-supervised Imitation Learning (SIL) method to explore unseen environments by imitating their own good decisions in the past. The authors also show that SIL can derive better and more efficient strategies, which greatly reduces the difference in the success rate of agents in seen and unseen environments.

Chiang et al. [165] proposed a robot autonomous control algorithm that combines Deep RL and long-distance motion planning in 2018, which has strong adaptive capabilities. First, the local planning agent they trained can perform basic navigation actions and safely traverse short terrain without colliding with other moving objects. These local planners can accept input from noisy sensors. For example, 2D LiDAR data can provide the distance to obstacles, and the planner can calculate the linear and angular speeds required for robot control. The authors use Automatic Reinforcement Learning (AutoRL) to train local plans in a simulated environment. Its function is to automatically search for RL feedback and the neural network architecture. This allows the local planner to migrate well to real robots and environments that have never been seen before. On this basis, Aleksandra Faust et al. [166] use a local planner based on RL and probabilistic roadmaps to complete long-distance robot navigation tasks. Next, Anthony Francis and his team [167] replaced the RL local planner with AutoRL training, which improved the performance of long-distance navigation. A Simultaneous Localization And Mapping (SLAM) map is also added to the system, with the robot performing synchronized positioning and map reconstruction during the navigation process. This can be used as a resource for probabilistic roadmaps reconstruction. Since the SLAM map is noisy, this change also compensates for the difference in performance between the robots in the simulated environment and the real environment due to different levels of noise. In fact, the success rate of navigation in a virtual environment is almost the same as the success rate of experiments on a real robot. Finally, the researchers also added distributed map construction, which greatly increased the maximum map size that the robot can support.

D.2.2 Object detection technology based on LiDAR

The output of a 3D LiDAR is a point cloud. Unlike the representation of image data in the computer, which usually encodes the spatial relationship between pixels, point cloud data is represented by a set of disordered data points. Therefore, two methods to pre-process such data before using the DL model have been proposed.

The first one is projecting the point cloud onto a two-dimensional plane. This method does not directly consider the three-dimensional point cloud data, but initially projects the point cloud to some specific perspectives and then processes them, such as the front view and the bird’s-eye view. At the same time, image information from the camera can also be used through fusion. The research of Volumetric CNNs [168] and Multi-view CNNs for 3D shape recognition [169] are examples of works using this method.

The second approach is dividing the point cloud data into Voxels with spatial dependence. This method divides the three-dimensional space, introduces spatial dependency into the point cloud data, and then uses 3D convolution and other methods for processing. The accuracy of this method depends on the fineness of the three-dimensional space segmentation, but the computational complexity of 3D convolution could be high. Voxnet [170] and the research of Wu et al. [171] are some methods using this idea.

Different from the aforementioned methods of pre-processing point cloud data, Charles Qi et al.[172] proposed a new type of DL model for processing point cloud raw data: PointNet. The authors also verified that it can be used for a variety of cognitive tasks, such as classification, semantic segmentation, and target recognition.

D.2.3 Related work in CARLA simulator

The CARLA (Car Learning to Act) simulator, developed by Dosovitskiy et al. [173] in 2017, is an open-source simulator that can simulate the real traffic environment, pedestrian behavior, several car sensors, etc. Sauer et al. [174] used CARLA to estimate several affordances from sensor inputs to drive a car in a simulated urban environment. Muller et al. [175] developed a system using CARLA by training a driving policy from a scene segmentation network to output high-level control, thereby enabling transfer learning to the real world using a different segmentation network trained on real data. Pan et al. [128] achieved the transfer of an agent trained in simulation to the real world using a learned intermediate scene labeling representation. Reinforcement learning may also be used in a simulator to train drivers on difficult interactive tasks such as merging, which require a lot of exploration, as shown in Shalev-Shwartz et al. [176] A CNN operating on a space-time volume of bird's eye-view representations is employed by Luo et al. [177]; Lee et al. [178] completed the tasks like 3D detection, tracking, and motion forecasting. Finally, there exists a large volume of works on vehicle motion planning outside the Machine Learning context using CARLA, and Paden et al. [179] presented a notable survey.

D.3 Methodology

D.3.1 Reinforcement Learning

The main reference for this sub-section on RL is Sutton and Barto [46].

Markov Decision Process

Reinforcement Learning is one of the paradigms of Machine Learning, which is used to describe the agent-environment interaction and to solve tasks through learning strategies to maximize returns and achieve the specific goal. The common model of RL is the Markov Decision Process (MDP) which formalizes the RL problem. The interaction between the agent and environment evolves as follows: at each time step t the agent takes an action A_t based on the current state S_t, the environment thus returns the corresponding reward R_{t+1}, and makes the agent transfer to the new state S_{t+1}. Actions are what the agent can do to affect the environment and its evolution. States are all the information useful to predict the future and make decisions. Rewards are scalar values indicating whether the behavior of the agent is good or bad toward solving the task. MDPs can be expressed as $M =< S, A, P, R, \gamma >$. Among them: $s \in S$: a limited set of states, s represents a specific state; $a \in A$: A limited set of actions, a represents a specific action; P is the state transition matrix based on the environment. Each item is the probability of transferring to a state s' after taking the action a starting from the state s, expressed as $P\left(s_{t+1} = s' \mid s_t = s, a_t = a\right)$; R is the reward function that maps state-action pairs to rewards, expressed as $R\left(s_t = s, a_t = a\right)$; γ is the discount with the value range [0, 1] which indicates the weight given to immediate and future rewards. With $\gamma = 0$ the agent considers only the immediate reward at the current time, while with γ tending to 1 it considers future rewards more. From the MDPs formalization, one can notice that they follow the Markov property: state transitions depend entirely on the present state and the selected action, thus states do not depend on their history.

Value Iteration

We want the agent to learn a *policy* i.e., a strategy. A policy is a map from states to the probabilities to select an action. Specifically, we are searching for the optimal policy which is the one that maximizes the *cumulative reward* (the discounted sum of rewards over time) defined as: $G_t = \sum_{t=0}^{n} \gamma^t r_{t+1}, \quad 0 \leq \gamma < 1$. A way to produce a policy is to estimate the *value function* $v_\pi(s)$. Given the function, the agent needs only to perform the action with the greatest value. The value function is a function that associates at each state a value indicating the cumulative reward we expect to gain, namely the *expected return* $\mathbb{E}[G_t]$, by following a policy π starting from such state. Similarly, the *action-value function* $q_\pi(s, a)$ associates to state-action pairs the expected return performing a starting in s and following π thereafter. The value function is defined as: $v_\pi(s) = \mathbb{E}[G_t \mid s_t = s]$. The action-value function is defined as: $q_\pi(s, a) = \mathbb{E}[G_t \mid s_t = s, a_t = a]$. Supposing we completely know the problem's MDP including the state transition matrix, we could use the Bellman optimality equation to compute the optimal action corresponding to the current state by solving a system of nonlinear equations. The equations are as follows for the value and action-value respectively:

$$\begin{aligned} v_*(s) &= \max_a \textstyle\sum_{s',r} p(s', r \mid s, a)\, [r + \gamma v_*(s')] \\ q_*(s,a) &= \textstyle\sum_{s',r} p(s', r \mid s, a)\, [r + \gamma \max_{a'} q_*(s', a')] \end{aligned} \tag{D.1}$$

As we typically do not know the state transition matrix and the reward distribution, we can instead estimate the action-value function based on experience, known as the Monte Carlo approach.

Q-learning

Q-learning is an algorithm that estimates the action-value function. At every iteration, the estimation is refined thanks to the new experience, and every time the evaluation becomes more precise, the policy gets closer to the optimal one. In Q-learning, a table associates state-action pairs to a value randomly initialized. At every new experience of the agent ($s_t, a_t, r_{t+1}, s_{t+1}$ sequence) the error made at time t by the estimation is computed and used to update the table. The update function, where α is the learning rate, is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{D.2}$$

When the number of states increases, it becomes impossible to use a table to store the Q-function. Deep Q-Network uses a neural network to approximate the Q-function. Other than solving the table dimension problem, the NN also provides better generalization and the ability to deal with non-Markovian environments as the experience gained in one state is useful also to similar ones, unlike the table that requires perfect matching. This research uses DQN as the RL algorithm to train cars to realize AD.

D.3.2 PointNet Algorithm

The output format of 3D-LiDAR is point cloud data. Point cloud data is a subset of points in Euclidean space, and it has the following three characteristics [172]:

- Disorder: point cloud data is a collection, and it is not concerned with the data order. This means that the model for processing point cloud data needs to be invariant to the different arrangements of the data. The methods used in the literature include: (i) reordering the disordered data; (ii) using all arrangements of the data followed by a Recurrent Neural Network; (iii) using symmetric functions. Due to the simplicity of the third method and easy implementation in the model, PointNet uses it with the symmetric function of max-pooling to extract the features of the point cloud data.
- Spatial relationship between points: an object is usually composed of a certain number of points in a specific space, which means that there is a spatial relationship between these points. To make effective use of this correlation, the algorithm used concatenates local and global features to aggregate information.
- Immutability: the target represented by the point cloud data should be invariant to spatial transformations, such as rotation and translation. Before performing feature extraction, the algorithm first aligns the point cloud data to ensure invariance. The alignment operation is achieved by training a small network to obtain the transformation matrix and multiplying the input point cloud data.

This research uses the PointNet model to process the 3D LiDAR data. The key processes of PointNet are [172]:

- Enter a collection of all point data of a frame, expressed as a 2D tensor of $N \times 3$, where n represents the number of points, and 3 corresponds to x, y, z coordinates.
- The input data is aligned by multiplying with a transformation matrix learned by T-Net, which ensures the invariance of the model to specific spatial transformations.
- After performing feature extraction on each point cloud through multiple Muti-Layer Perception (MLP), a T-Net is used to align the features.
- Perform max-pooling operation to get the final global feature.
- For the classification task, the global feature is used to predict the final classification score through MLP; for the segmentation task, the global feature is concatenated with the local features of each point cloud learned before, and then the classification result of each point is obtained through MLP.

D.3.3 Carla Simulation Environment

Training the car in a real environment may cause damage and bring significant costs. So, the debugging and effect evaluation of AD algorithms should be done in a hyper-realistic simulation environment firstly. This research selects CARLA36 as the simulation environment for AD development and testing. The simulator embeds the 3D urban scene and supports perception, planning, and control. CARLA is open source, grounded on C++ and Unreal Engine, and relies on a client-server structure. The simulation itself runs on the server. On the client-side, the user can interact with the agents controlling the scene settings. It uses OpenDrive[180] standard to define different roads and map types, trying to replicate the complexity of urban environments. CARLA can create countless events and agents providing them with a wide range of sensors (including 3D LiDAR as shown in Fig. D.2) through powerful Python API which meets the requirements of this research. It can thus reproduce common scenes in the environment, such as lane conditions, road conditions, obstacle distribution, weather, pedestrian behaviors, etc.

CARLA provides three approaches to AD systems: (i) classical modular method, including vision-based perception modules, rule-based planners, and behavior controllers; (ii) end-to-end Imitation Learning method; (iii) end-to-end Reinforcement Learning method. CARLA is particularly well-suited to RL applications: it can simulate various kinds of emergencies, in which RL algorithms define specific rewards according to vehicles' responses. Therefore, as long as the simulator can simulate enough emergencies, the RL algorithm can learn the corresponding processing methods, instead of defining separate rules for each situation. Moreover, the simulator can enhance the learning efficiency, producing emergencies that cannot be solved by the current RL algorithm.

D.4 Experiment

This research explores how *raw* 3D LiDAR data can be used in conjunction with RL to address AD tasks. Specifically, as this is the first application of RL to 3D LiDAR points, we started with a simple setting including a simple task as well as a reduced point set and action set. The task is lane following in an urban town featuring complex roadways with intersections and roundabouts. An example of the scene is shown in Fig. D.2. The environment is simulated using CARLA as it allows to simulate 3D LiDARs in urban environments. The CARLA environment and the RL environment work asynchronously with the CARLA simulation independently updated at fixed time intervals. The agent thus uses the latest available measurements without waiting for updates while the simulation performs the latest action delivered by the agent. The RL environment is structured as follows. The state is composed of two subsequent raw 3D LiDAR measurements with 2048 points each. The points are split on an 8° vertical FoV and 270° horizontal FoV, with a vision range of 50m. Having two subsequent measurements allows us to include the notion of movement into the state. The actions are discrete due to the DQN nature and are forward, right, left. The agent actions affect only the steering while the car's throttle is kept constant. The reward function gives a positive reward (0.01) each time the forward action is selected. When the car collides or leaves the lane a negative reward (-1) is given, and the episode ends. The new episode starts with the car in a random spawn point. The agent aims to maximize the cumulative reward by following the street as long as possible.

The RL agent software is composed of *two* main parts: **the training cycle** and the NN. The training cycle is the typical one of DQN with a ϵ-greedy policy for exploration, replay buffer for I.I.D. data, and target network to reduce instability. The selected **Neural Network** should be designed to extract features from point clouds to successfully support the DQN algorithm. We considered and compared two NNs to this aim: PointNet and 1D CNN, as both can extract the spatial relation from data. PointNet is specifically designed for point cloud processing and can deal with the disorder and transformations of the points [172]. CNNs are known to perform better on structured and spatially related data with local information, and 1D CNNs have been proved to perform well on 2D LiDAR data [3]. The input to the networks is a vector of [2048, 6] values, with 2048 being the number of points and 6 are the coordinates x_t, y_t, z_t at time t and $x_{t-1}, y_{t-1}, z_{t-1}$ at time $t-1$. The output of the NNs is the expected return for each of the 3 actions. We tested multiple settings for each network:

- The best one for PointNet is a reduced version composed of a (reduced) T-Net with two convolution layers with 32 and 64 filters respectively and kernel size 1; a global max-pooling layer; a dense layer with 64 units; a dense layer with 36 (*num_features*num_features*) units with orthogonal regularizer and initialized to zero. The T-Net is followed by a convolution layer with 32 filters and kernel size 1; a global max-pooling layer; a dense layer with 64 units; a dense layer with 3 units as output layer. Every layer uses the Batch Normalization with momentum 0.

- The best for 1D CNN instead contains the following hidden layers: a convolution layer with 32 filters and kernel size 4; another convolution layer with 64 filters and kernel size 2; a global max-pooling layer; a dense layer with 64 units. The complete structure of the NNs is shown in Fig. D.3.

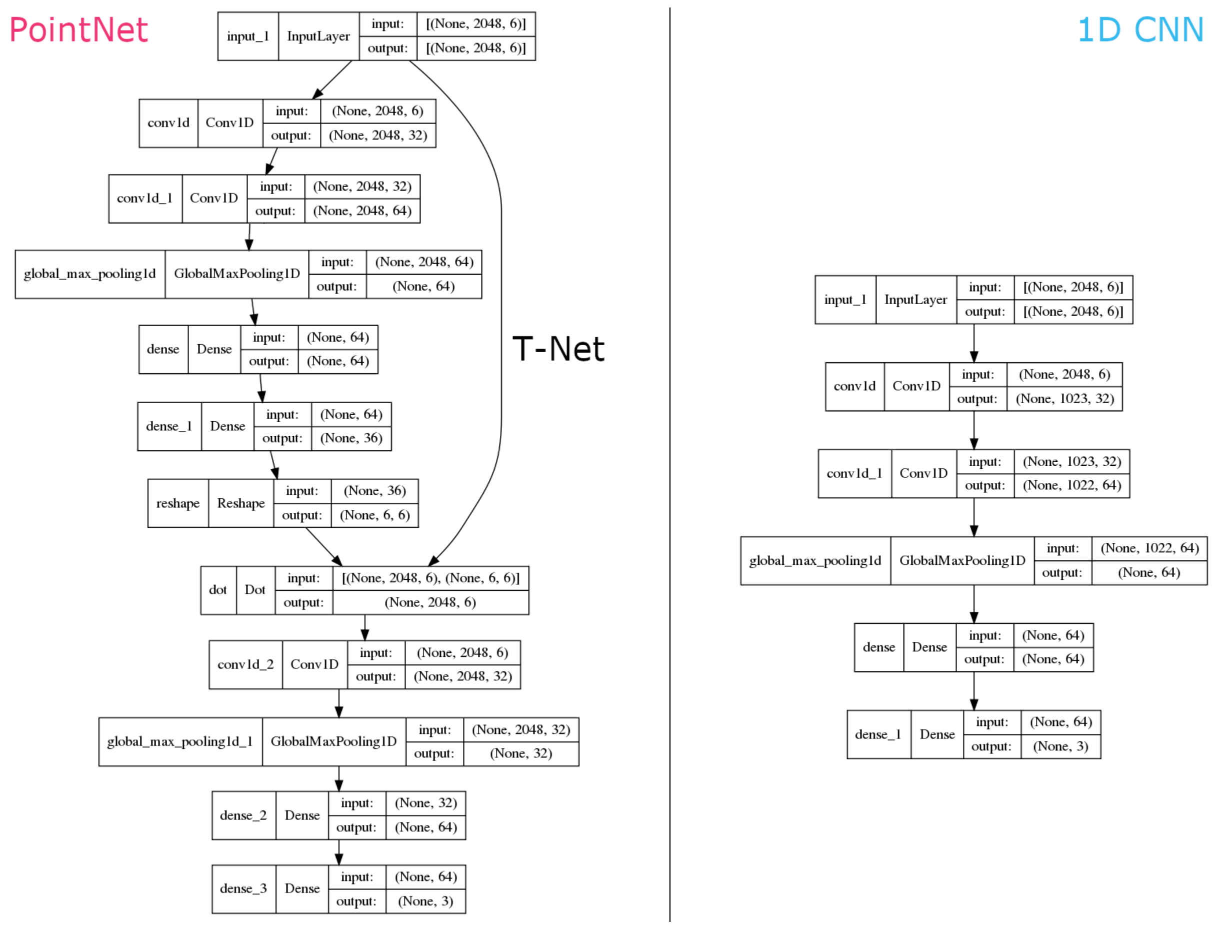

PointNet
input_1 | InputLayer | input: [(None, 2048, 6)] | output: [(None, 2048, 6)]
conv1d | Conv1D | input: (None, 2048, 6) | output: (None, 2048, 32)
conv1d_1 | Conv1D | input: (None, 2048, 32) | output: (None, 2048, 64)
global_max_pooling1d | GlobalMaxPooling1D | input: (None, 2048, 64) | output: (None, 64)
dense | Dense | input: (None, 64) | output: (None, 64)
dense_1 | Dense | input: (None, 64) | output: (None, 36)
reshape | Reshape | input: (None, 36) | output: (None, 6, 6)
T-Net
dot | Dot | input: [(None, 2048, 6), (None, 6, 6)] | output: (None, 2048, 6)
conv1d_2 | Conv1D | input: (None, 2048, 6) | output: (None, 2048, 32)
global_max_pooling1d_1 | GlobalMaxPooling1D | input: (None, 2048, 32) | output: (None, 32)
dense_2 | Dense | input: (None, 32) | output: (None, 64)
dense_3 | Dense | input: (None, 64) | output: (None, 3)
1D CNN
input_1 | InputLayer | input: [(None, 2048, 6)] | output: [(None, 2048, 6)]
conv1d | Conv1D | input: (None, 2048, 6) | output: (None, 1023, 32)
conv1d_1 | Conv1D | input: (None, 1023, 32) | output: (None, 1022, 64)
global_max_pooling1d | GlobalMaxPooling1D | input: (None, 1022, 64) | output: (None, 64)
dense | Dense | input: (None, 64) | output: (None, 64)
dense_1 | Dense | input: (None, 64) | output: (None, 3)

Figure D.3: Visual representation of the two NN models (PointNet, left; 1D CNN, right)

Our preliminary results show that DQN with both the considered NNs can learn how to follow the street in the CARLA simulator using 3D LiDAR data. However, the agent failed to fully master the task as it misbehaves on multiple occasions causing crashes. As a result, there is room for improvement and further research must be performed to assess the suitability of the approach. The ability of the NNs to learn is proved by the loss curve shown in Fig. D.4 (left) and the RL agent learning is evident by the plot displayed in Fig. D.4 (right) that shows the average cumulative reward on 100 episodes of the two NNs. The agent was trained for 5 million and 7.5 million steps for the PointNet and 1D CNN, respectively. The two networks have comparable results, but the 1D CNN performed slightly better. This may be due to the simpler structure of the 1D CNN that can better adapt to a not too complex problem like line following.

The source code of the project is available at: `github.com/MichaelBosello/f1tenth-RL`

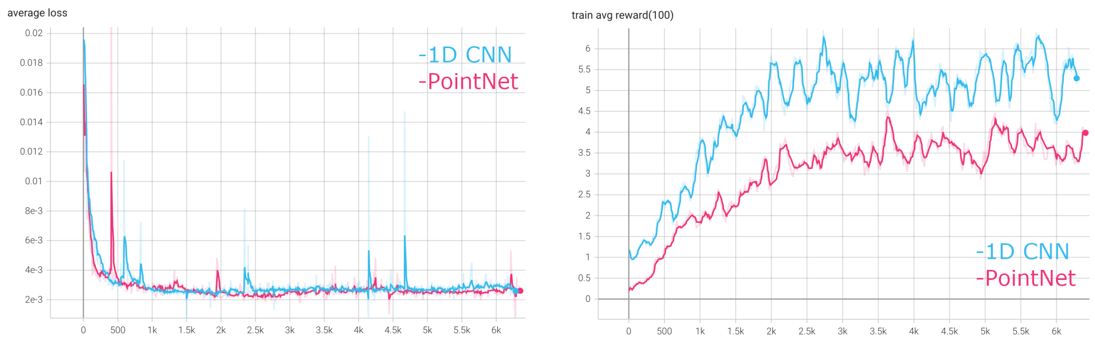


Figure D.4: NNs training loss during experiments (left). Average cumulative reward on 100 episodes in the training phase (right).

D.5 Conclusions

World Health Organization (WHO) estimated that road traffic crashes cost most countries 3% of their gross domestic product [181]. AD technology cannot only improve driving safety but also improve the efficiency of the entire transportation system; upgrading it to the next level is essential for the vehicle industry and has great significance to our society. This research aims at harnessing the power of Reinforcement Learning to interpret complex data from 3D LiDAR sensors. 3D LiDAR maps the surroundings in all three dimensions with a fine time granularity (i.e., milliseconds); this provides a massive amount of data for in-vehicle processing. This data is highly dynamic and, potentially, represents all the countless driving scenarios that can ever happen. In contrast with supervised learning, RL can adapt to cope with new cases thanks to simulation and has the potential to augment current AD by introducing causality based on a reward system designed to prize good choices and penalize bad ones. This project will be implemented via hybrid simulations: the initial phase will be performed in a simulated environment to increase the learning speed and avoid hardware disruptions. Once the system reaches stability, it will be ported on a model car. This will re-introduce the noise – typical from hardware tolerances – and will test our conjectures on RL resilience to new scenarios. This approach can also reduce the cost of Machine Learning by ensuring high-precision views. The results of this project will play a positive role in promoting the field of AD.

Acknowledgments

This work was supported in part by the Macao Polytechnic University – Edge Sensing and Computing: Enabling Human-centric (Sustainable) Smart Cities (RP/ESCA-01/2020). This work has received funding from the LiBER project under POR FESR programme by Emilia Romagna Region, years 2019-2022.

Appendix E

Race Against the Machine: a Fully-annotated, Open-design Dataset of Autonomous and Piloted High-speed Flight

Reprinted, with permission, from:

M. Bosello, D. Aguiari, Y. Keuter, *et al.*, “Race against the machine: A fully-annotated, open-design dataset of autonomous and piloted high-speed flight,” *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3799–3806, 2024. DOI: 10.1109/LRA.2024.3371288

Race Against the Machine: a Fully-annotated, Open-design Dataset of Autonomous and Piloted High-speed Flight

Michael Bosello, Davide Aguiari, Yvo Keuter, Enrico Pallotta, Sara Kiade, Gyordan Caminati, Flavio Pinzarrone, Junaid Halepota, Jacopo Panerati, and Giovanni Pau

Abstract – Unmanned aerial vehicles, and multi-rotors in particular, can now perform dexterous tasks in impervious environments, from infrastructure monitoring to emergency deliveries. Autonomous drone racing has emerged as an ideal benchmark to develop and evaluate these capabilities. Its challenges include accurate and robust visual-inertial odometry during aggressive maneuvers, complex aerodynamics, and constrained computational resources. As researchers increasingly channel their efforts into it, they also need the tools to timely and equitably compare their results and advances. With this dataset, we want to *(i)* support the development of new methods and *(ii)* establish quantitative comparisons for approaches originating from the broader robotics and artificial intelligence communities. We want to provide a one-stop resource that is comprehensive of *(i)* aggressive autonomous and piloted flight, *(ii)* high-resolution, high-frequency visual, inertial, and motion capture data, *(iii)* commands and control inputs, *(iv)* multiple light settings, and *(v)* corner-level labeling of drone racing gates. We also release the complete specifications to recreate our flight platform, using commercial off-the-shelf components and the open-source flight controller Betaflight, to democratize drone racing research. Our dataset, open-source scripts, and drone design are available at: `github.com/tii-racing/drone-racing-dataset`.

E.1 Introduction

Unmanned aerial vehicles (UAVs) and multi-rotor drones have become ubiquitous robotic platforms, supporting a wide array of industries from video-making to warehouse monitoring, to surveillance and inspection of energy and transport infrastructure. Drone racing, in particular, has emerged as the go-to benchmark problem to measure the advances made by researchers in the quest to surpass human-level, autonomous performance in fast and aggressive flight [14], [49], [182], [183]. Yet, drone racing competitions, equipment, and venues can still be difficult and expensive to access.

The last decade of machine learning progress has shown how datasets, open standards, and open-source code help scientific progress and transparency, shaping the entire scientific fields [184]. One of the fundamental advantages of datasets is to greatly simplify and shorten the development pipeline of new methods by allowing researchers to re-use the tried and tested data collection and consolidation work of others. Datasets allow researchers from different parts of the world and disciplines to work on common problems.

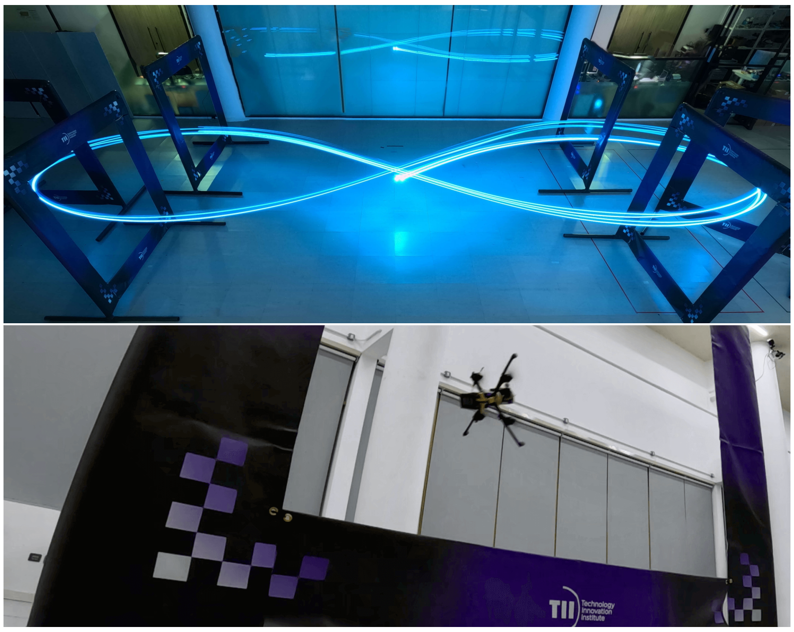


Figure E.1: Long exposure, low-light capture of the open-design racing drone used to collect the dataset (top), and an aggressive maneuver through one of the labeled gates (bottom).

Today, several conferences and journals, including NeurIPS, the IEEE Robotics and Automation Letter, and the International Journal of Robotics Research explicitly solicit data and benchmark papers as a way to increase the number and visibility of peer-reviewed datasets [185].

Robotics datasets and benchmarks are as important and beneficial to the community as they are challenging to create—because of the idiosyncrasies of robotic hardware and real-world systems. Notable robotics datasets have focused on vision problems, e.g., the KITTI Vision Benchmark Suite [186], including data from stereo cameras, GPS, and laser scanners for tasks such as object detection, tracking, and visual-inertial odometry (VIO) [187], or the use of special, novel sensors and instruments [188]. Early drone racing datasets [189] also focused on scene understanding and gate pose estimation problems, while more recent datasets have put a greater emphasis on the coupling with on-board inertial data, ground truth information, and controls [9], [190].

Our dataset builds upon these and aims to be a one-stop resource for researchers to simultaneously pursue multiple lines of work, including semantic scene understanding, VIO, mapping and planning, and data-based system identification for fast and aggressive multi-rotor flight. As for a benchmark to be successful, it must be effectively and easily repeatable, in Section E.3, we release the complete design specifications of the drone used to collect the dataset.

The main contributions of our work are as follows.

- The public release of a dataset for drone racing (inclusive of open-source code for visualization and post-processing) that is characterized by:
  - fast (>20m/s), aggressive flight, both autonomous and human-piloted, on multiple trajectories (including a complex 3D racing track);
  - high-resolution, high-frequency (~10^2Hz) collection of visual, inertial, and motion capture data;
  - versatility—our dataset includes drone racing gates fully labeled to the level of individual corners [191] (for VIO, self-localization, scene understanding, etc.), information about commands, control inputs, and battery voltages (for estimation problems, etc.), as well as lighting and sensor settings metadata.
- The open design (with commercial off-the-shelf (COTS) components) of the racing drone used to collect the data. For direct comparisons, the same design allows, without modifications, both autonomous and piloted flight.

E.2 Related Work

In Table E.1, we summarize the—all very recent—datasets for vision-based flight, drone racing, and aggressive quadrotor control related to our own. Earlier datasets for multi-rotor VIO and simultaneous localization and mapping (SLAM) included the 2016 EuRoC [192] and the 2017 Zurich Urban [193] micro aerial vehicle (MAV) datasets. However, these were characterized by comparatively lower speeds and frequencies of images and collected data than those needed for drone racing.

The last five years have seen a renewed interest in aggressive flight [9]. In [187], a new dataset was introduced to validate stereo VIO methods for fast autonomous flight, although without the inclusion of racing gates. The Blackbird dataset [50] was proposed as an aggressive indoor flight dataset for agile perception. While inertial data were collected in the real world, its high-resolution images were generated in simulation. In 2019, Lockheed Martin released an image dataset for Test#2 of its AlphaPilot challenge's virtual qualifiers [189], that did not include drone state information (while Test#3 consisted of control in simulation).

The work closest to ours is the dataset presented in [49]. It also observes that previous benchmarks for drone VIO had focused on too slow trajectories. Our dataset differs from [49] in that it provides higher-frequency RGB mono-camera images (the information used by human pilots), it includes piloted and autonomous flights on identical tracks, and it is fully annotated with high-frequency motion capture data as well as the gates' corner labels.

Other recent, specialized datasets for drone racing and aggressive multi-rotor flight include [194], [195]. Another open-source, open-hardware racing drone was proposed in [196]. In comparison, our design has a more powerful companion computer and it can be seamlessly used by human pilots.

Compared to the existing literature (Table E.1) [9], our dataset *(i)* includes high-resolution, high-frequency images captured in different light settings and *(ii)* it is fully annotated, down to the gates' individual corners. It can support research in all aspects of autonomous drone racing from VIO, to gate pose estimation [197], [198] and data-driven control.

Table E.1: Comparison of multi-rotor and drone racing datasets for visual-inertial odometry, scene understanding, and control

Ref.	Time &	Data	Conditions		Gates		Top	Vision/Camera Specifications					Pose/Inertial Data		Control Inputs		Battery	Data
	Distance	Coll.	Scene	Lighting	Pose	Labels	Speed	Resolution/Freq.	Color	FoV	Stereo	Event	IMU	MoCap	CTBR	Motor	Voltage	Formats
Ours TII-RATM	~29' ~7km	Real	Indoor	3 Levels Labeled	✓	✓†	9.5m/s$^{\diamond}$ 21.8m/s¶	640x480@120Hz	RGB	D 175°	✗	✗	@500Hz	@275Hz	@100Hz	@100Hz	@50Hz	`rosbag`, CSV, JPEG
[49] UZH-FPV	~24' ~11km	Real	Indoor; Outdoor	Multiple, Unlabel.	✗	✗	26.8m/s¶ 23.4m/s$^{\diamond}$	848x800@30Hz 346x260@50Hz 640x480@30Hz*	Grayscale Grayscale Grayscale	D 163° 120° 186°	✓	✓	@200Hz @500Hz @1000Hz	@20Hz§	✗	✗	✗	`rosbag`, TXT, PNG
[189] AlphaPilot	n/a‡	Real	Indoor, 1 Gate	Multiple, Unlabel.	✗	✓$^{\|\|}$	n/a	1296x864	RGB	n/a	✗	✗	✗	✗	✗	✗	✗	JSON, JPEG
[50] Blackbird	~10h ~100km	Real + Synth.	Indoor, 5 Scenes	Multiple, Unlabel.	✗	✗	7m/s	1024x768@120Hz†† 1024x768@360Hz††	Grayscale RGB	V 60°	✓	✗	@100Hz	@360Hz	✗	@190Hz	✗	`rosbag`, CSV, MP4, PNG Depth
[192] EuRoC	~22' ~1km	Real	Indoor, 2 Scenes	Multiple, Unlabel.	✗	✗	2.3m/s	752x480@20Hz	Grayscale	H 115°	✓	✗	@200Hz	@20Hz§ @100Hz	✗	✗	✗	CSV, PLY, PNG
[187] GRASP	~10' ~3km	Real	Outdoor; 1 Scene	Multiple, Unlabel.	✗	✗	17.5m/s	960x800@40Hz	Grayscale	n/a	✓	✗	@200Hz	✗	✗	✗	✗	`rosbag`
[194] EyeGaze	~300' ~100km	Synth.	Indoor, 2 Scenes	Multiple, Unlabel.	✓	✓¶¶	13.8m/s	800x600@60Hz	RGB	120°	✗	✗	✗	@500Hz**	@500Hz	✗	✗	CSV, MP4
[195] NeuroBEM	~75'	Real	n/a	✗	✗	✗	18m/s	✗	✗	✗	✗	✗	@1000Hz	@400Hz	✗	@1000Hz	@400Hz	CSV

†Bounding boxes, top-bottom left-rigth corners. $^{\diamond}$Piloted. ¶Autonomous. *Stereo.
§Leica laser tracker. ‡9300 frames. $^{\|\|}$Internal corners. ††Synthetic camera images.
¶¶Area of interest of the gaze. **Simulated.

E.3 An Open-design Quadrotor for Autonomous Drone Racing and Data Collection

To collect this dataset, we designed a new custom quadrotor (Figure E.2). Our design is based on a 5" carbon-fiber frame with a propeller-to-propeller diagonal of 215mm. The fully-assembled drone, including the battery, weighs ~870g, has a thrust-to-weight ratio of 7.5 (load cell), and can reach a maximum speed (measured outdoors) of 179km/h—allowing for the aggressive maneuvers required in drone racing. The top linear acceleration and angular velocity in the dataset are 69.85m/s^2 (>7g's) and 20.01rad/s. Importantly, our design can be used, without modifications, as both an autonomous and a human-piloted FPV racing drone. We do this to create a true test bench to benchmark drone racing autonomy against human performance. Our dataset includes both autonomous and piloted flights. Its components are listed in Table E.2.

E.3.1 Design Overview

Table E.2: Off-the-shelf components needed to re-create the open-design racing drone used to collect the dataset

Component	Producer	Model
ESC	T-MOTOR	F55A PROII 6S 4IN1
FCU	Holybro	Kakute H7 v1.3
RC Receiver	Team BlackSheep	CROSSFIRE NANO RX (SE) LONG RANGE
Battery	Tattu	R-Line v5.0 1400mAh 22.2V 150C 6S1P LiPo
Computer	NVIDIA	Orin NX 16GB Module
Carrier board	Seeed Studio	A203 (Version 2)
BEC	Matek	BEC12S-PRO
Camera	Arducam	B0179 8MP IMX219
FPV Camera	Foxeer	T-Rex Mini 1500TVL
Frame	Pyrodrone	Hyperlite Floss 3.0 Race Frame 5"
Motors	T-MOTOR	F60 PRO V 2020KV
Propellers	T-MOTOR	T5147

The quadrotor has three main sub-systems: *(i)* the quadrotor electronics, *(ii)* the autonomous module, and *(iii)* the First-Person-View (FPV) system. These sub-systems are combined by means of the frame and fasteners. The assembled system comprises two cameras. One digital, connected to the autonomous module, and one analog, used by the human pilot in the FPV system. The two cameras share the same mount, and the FPV camera is placed above the digital one (Figure E.2).

Quadrotor electronics

These are *(i)* the electronic speed controller (ESC), *(ii)* the Kakute H7 v1 flight controller unit (FCU), *(iii)* the radio controller (RC) receiver, and *(iv)* the battery. They are mounted underneath the frame. These components are protected by the aluminum standoffs connecting the frame and a 3D-printed custom battery cage. The FCU hosts an STM32H7 microcontroller and it is capable of running multiple firmware, including Betaflight, Ardupilot, and PX4.

Autonomous module

It comprises *(i)* an NVIDIA Orin NX (hosted on the A203v2 carrier board with SSD and wireless card), *(ii)* the battery eliminator circuit (BEC) powering it, and *(iii)* an Arducam RGB camera. These components are placed above the frame and are secured by two 3D-printed plates, connected by aluminum standoffs. The top plate provides the mount for the cameras. A MIPI CSI-2 ribbon cable connects the companion board with the Arducam. The FC is connected via a serial port, using a shielded cable. This connection is used to both control the drone and read FC's sensors.

FPV system

Independent from the autonomous module and used for human piloting instead, it comprises an FPV analog camera, a video transmitter, and its antennas, all placed above the frame.

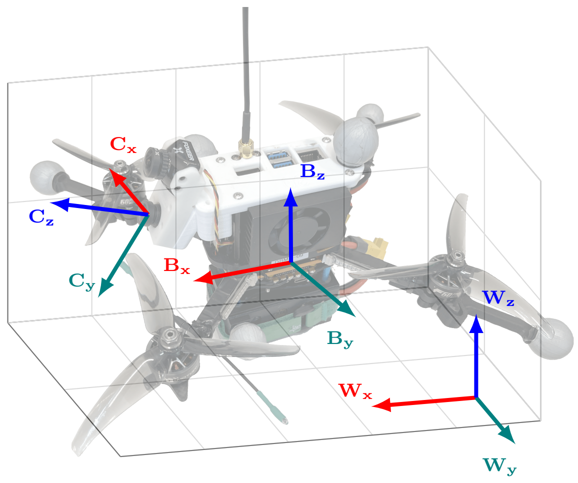


Figure E.2: The drone platform used to record the dataset, the body frame *B* has its origin at the FCU's IMU location, the camera frame *C* is located where the bottom lens is (the top lens being the one of the FPV system).

The CAD models of all the 3D printed parts and the bill of materials of all other COTS components are available online[1], with a video tutorial on how to re-create our drone.

E.3.2 Sensors

Our quadrotor is equipped with multiple sensors for autonomous and piloted aggressive flight:

InvenSense MPU6000 IMU

Part of the quadrotor electronics (E.3.1), it is embedded into the FC. This module has two functions: delivering precise, real-time tri-axis angular rate sensor (gyroscope) data, as well as accurate tri-axis accelerometer data. The raw IMU data are read by the companion computer in a demand/response exchange fashion, using the Multiwii Serial Protocol (MSP) [199].

Arducam B0179 IMX219 8MP RGB Bayer camera

Part of the autonomous module (E.3.1), it captures 640×480 pixel frames at 120Hz with a diagonal field-of-view (FOV) of 175° and a horizontal field-of-view (HFOV) of 155°. Its readout speed is 3.22×10^{-5}s, computed as the line length divided by the pixel rate before re-scaling, i.e.,

[1]`github.com/tii-racing/drone-racing-dataset/tree/main/quadrotor`

3560px/(1280×720×120Hz). It is one of the most used lightweight embedded cameras on the market and it is fully supported by NVIDIA with dedicated MIPI CSI-2 drivers. The image YUV frames are captured in NV12 format. In the NV12 format, each pixel in the luminance (Y) component is represented by a single byte. The chrominance (UV) components are interleaved and share memory locations. The image is then converted to BGR, a common color format suitable for processing and analysis. Eventually, the image is saved as JPEG. This pipeline is accomplished by the NVIDIA GStreamer plugin on the NVIDIA companion computer.

Foxeer T-Rex Mini 1500TVL

It is the low latency (6ms) camera in the FPV system (E.3.1). For the sake of the data collection in this letter, this was used by a human pilot in conjunction with a pair of 1280x960 OLED Fat Shark HDO2 googles.

E.3.3 Software

Quadrotor electronics (E.3.1) software

The FC firmware we used is Betaflight 4.3.1 [200], whose proportional-integral-derivative controller (PID) was tuned by a human pilot. The companion computer uses MSP to both send commands to the FC and read its sensors. Accordingly, we activate Betaflight's `MSP_OVERRIDE` feature to bypass the RC controller's commands. An `MSP_OVERRIDE` channels mask limits to override the motor commands: for safety reasons, a human can always disarm the drone with an RC controller.

Autonomous module (E.3.1) software

On the Orin NX module, we installed NVIDIA JetPack 5.1.1. The JetPack includes Jetson Linux 35.3.1 Board Support Package (BSP) with Linux Real-Time Kernel 5.10, an Ubuntu 20.04-based root file system with CUDA 11.4 support. We use the Humble (current LTS) distribution of the Robot Operating System 2 (ROS2) as the middleware for communication between the perception, planning, and control modules.

E.4 Data Collection Protocol

E.4.1 Flight Arena, Racing Gates, and Motion Capture System

Our dataset was recorded in a 25 (L) by 9.7 (W) by 7 (H) meters indoor flying arena. The arena is equipped with a 32-camera Arqus A12 Qualisys Motion Capture (MoCap) system, tracking 6DoF poses of defined rigid bodies with millimeter accuracy at 275Hz. The drone design from Section E.3 is equipped with six 25mm markers defining a single rigid body. The markers are mounted on the top plate, battery cage, and 48.2mm arm extensions, preventing the markers from being occluded by the propellers (Figure E.2). The origin of the drone's rigid body was placed at the location of the FCU's IMU, as shown in Figure E.2. The IMU was also calibrated using the RC before each take-off [200].

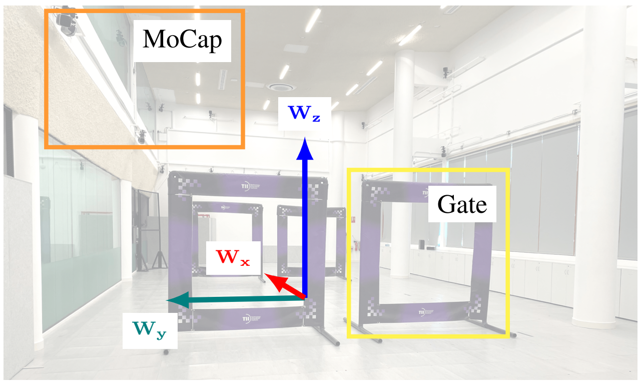


Figure E.3: The 25×9.7×7 meters indoor arena, instrumented with 32 Qualisys MoCap cameras and equipped with four 5×5 feet racing gates used to record the dataset.

We use a minimum of 4 and up to 7 gates to create 2D and 3D racing tracks. The 7-gate track is inspired by the one in [14] but shrunken to meet our arena size constraints (Figure E.3). This track features challenging maneuvers like the Split-S and sharp turns, and it evolves over the z-axis for 5 meters. The gates are made of PVC pipes covered by printed fabric banners. They measure 7 by 7 feet (213.36 cm) and have an internal opening of 5 by 5 feet (152.4 cm), similar to those used in major drone racing leagues [201]. Each racing gate's rigid body is defined by four markers placed in its inner corners.

E.4.2 Flight Program

Our dataset contains a total of 30 flights (Table E.3): 12 *human-piloted* and 18 *autonomous* ones. In either case, two shapes—*ellipse* and *lemniscate* (Figure E.4)—have been executed 6 times. The 6 additional autonomous flights are collected on a 3D race track. The 6 repetitions correspond to the combination of 3 different illumination conditions and 2 camera settings.

Table E.3: Summary of the flights recorded in the dataset

Control	Shape	Top Speed	Time	Distance
Autonomous	Ellipse[†]	**21.83 m/s**	149.08 s	455.27 m
	Lemniscate[†]	10.22 m/s	155.08 s	359.63 m
	Race Track[‡]	21.39 m/s	278.05 s	1161.51 m
Piloted	Ellipse[§]	9.50 m/s	575.38 s	2586.3 m
	Lemniscate[§]	8.93 m/s	**593.63 s**	**2593.47 m**

[†]Flown twice in 6 flights (3 brightness × 2 camera settings). [‡]Flown three times in 6 flights (3 brightness × 2 camera settings). [§]Flown as many times as possible in 6 flights (3 brightness × 2 camera settings).

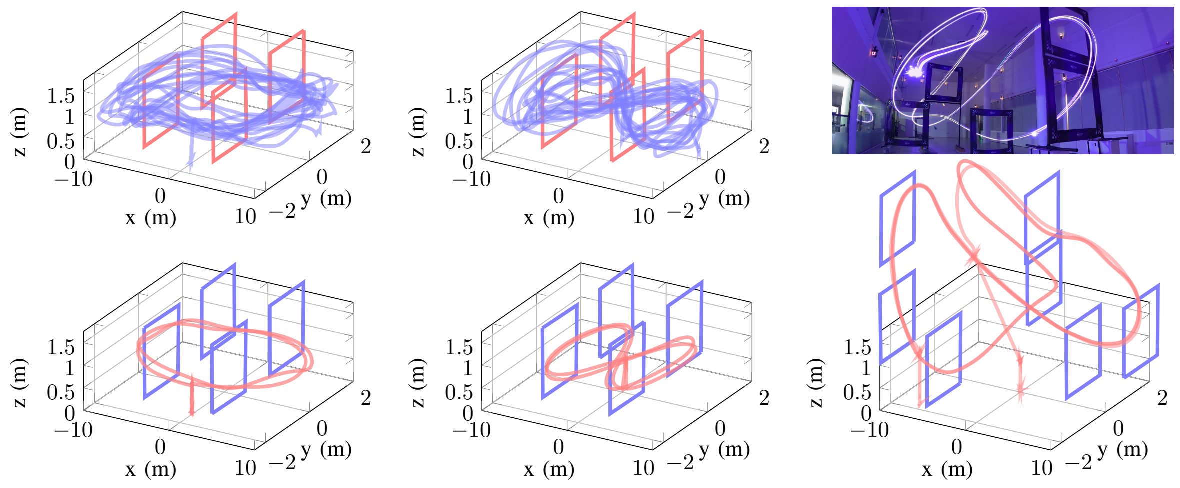


Figure E.4: Examples of recorded trajectories: piloted ellipse `flight-01p-ellipse` (top-left), piloted lemniscate `flight-07p-lemniscate` (top-center), autonomous ellipse `flight-01a-ellipse` (bottom-left), autonomous lemniscate `flight-07a-lemniscate` (bottom-center), and autonomous race track `flight-13a-trackRATM` (bottom-right).

Brightness levels

We created three levels of brightness for data collection. The *high* brightness level is achieved with both natural and artificial light (max./avg./min. measured illuminance in the arena of 2480 lx, 1500 lx, and 254 lx); the *medium* level has controlled light, and it is obtained by turning on all the artificial lights in the arena but keeping natural light out using the blinds (955 lx/672.3 lx/254 lx). The *low* brightness condition is obtained by turning off most of the lights in the arena and keeping the blinds down, (216 lx/72 lx/34.2 lx).

Camera settings

We used two different nvarguscamerasrc settings, *auto* exposure time and gains, and *fixed* exposure time (2.5ms), analog gain (2), and digital gain (1). In the *auto* setting, the image is brighter but suffers from higher motion blur. In *fixed*, the image is darker, with limited motion blur.

E.4.3 Human-piloted and Autonomous Control

In autonomous mode, each flight contains exactly two (ellipse, lemniscate) or three (race track) laps, lasting ~25 and ~45s, respectively. The human-piloted flights last between 84 and 108s, during which the pilot tries to achieve the maximum number of laps possible on a single battery charge. We did not clip the human-piloted flights as they exhibit higher variability and cover a larger state space.

In the autonomous flights, we use a Proportional Derivative (PD) controller based on [202] for the 2D tracks and a Model Predictive Controller (MPC) based on [203] for the 3D track. The motion capture system feeds the current quadrotor pose into the controller at 275Hz through WiFi. Control commands are sent to the FCU at the same update frequency of 275Hz by the PD controller, and 160Hz by the MPC.

A Python implementation of the PD controller and its tuned gains are available on GitHub[2].

The human pilot used a camera angle (for both their FPV camera and the recorded Arducam) of 30°. The autonomous runs were recorded with a camera angle of 40° for the lemniscate and 50° for the ellipse and race track, empirically chosen to maximize the gates' visibility. FPV pilots choose the camera angles according to the speed they plan to reach[3]. We used the same principle to choose the camera angle for the autonomous mode, making the gates visible at high speed.

E.4.4 Image Labeling

Often being the only well-known object in a racing environment, gates are key to relative localization and next-waypoint detection. In this dataset, we provide image labels in the form of bounding boxes and keypoints associated with the inner corners of all visible gates. Coupled with the drone's inertial data and the ground truth from the motion capture system, our dataset allows to reproduce and benchmark the state-of-the-art in gate pose estimation [204] as well as to develop new methods.

Labeling was first performed automatically, using a top-down keypoints detector [205], [206] trained on a synthetic dataset and fine-tuned on 5'000 manually labeled images. All the images were then labeled through an iterative process of manual review and re-training. Finally, all labels were eventually manually reviewed. Bounding boxes and corner positions are provided, based on the auto-labeling results and a human estimate, even for partially occluded gates. However, no distinction between occluded and fully visible keypoints is made. Thus, the only visibility values we provide are 0 (outside the image boundaries) and 2 (inside the image boundaries) as per COCO format definition [207]. As a convention, gates are not labeled when there are no visible corners.

E.4.5 Time Synchronization

We record three separate data streams during each flight: *(i)* a `rosbag` containing the FCU readings and the autonomous control setpoints, *(ii)* the on-board camera images, and *(iii)* the Qualisys motion capture measurements.

For the FCU data, we use custom ROS2 messages and the Real-Time Kernel to limit the jitter in the sensors' readings. Furthermore, the GStreamer pipeline allows us to save the images and timestamp them using the frame acquisition time.

In the end, two different clocks are involved: the real-time clock of the drone's companion computer, and the clock of the Qualisys workstation. Both were synced with a Network Time Protocol (NTP) server placed inside the facility, before each flight. The clock offsets w.r.t. the NTP server of the two machines were recorded before and after each flight to compute the jitter that occurred during the flight. The drone achieved a microsecond clock accuracy with Chrony, while the Qualisys workstation recorded a millisecond accuracy. The total jitter from start to end of a trajectory never exceeded 3ms.

E.4.6 Data Post-processing

The motion capture data were converted to CSV, and the clock offset with the onboard computer was removed. The ROS2 bags were also dumped into CSVs. All the data were then trimmed to remove

[2] `github.com/tii-racing/drone-racing-dataset/blob/main/scripts/reference_controller.py`

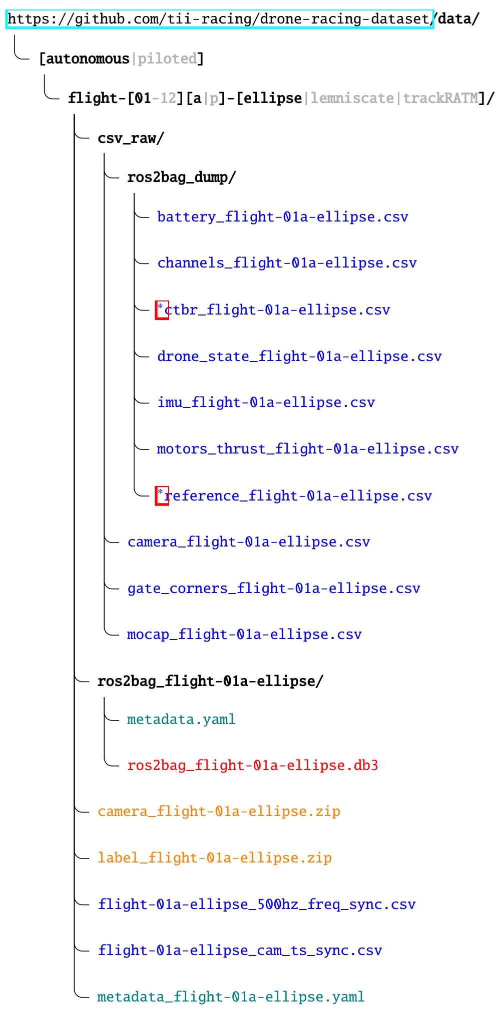
[3] `www.getfpv.com/learn/fpv-essentials/fpv-camera- angle-full-throttle-flight/`

pre-take-off and post-landing records. We used an open-source script for data alignment (Snipped E.6) to produce user-friendly comprehensive CSVs. The alignment is achieved through linear interpolation for all the fields with the exception of the rotation matrices, for which spherical linear interpolation [208] is used instead.

E.5 Data Format

For each flight, we recorded data from four different sources: *(i)* the motion capture system's ground truth (drone and gates poses), *(ii)* the Arducam IMX219 Bayer camera (images), *(iii)* the flight controller (IMU, battery, motors, RC), and *(iv)* companion computer (autonomous control reference and control inputs).

All data collected is timestamped (Unix epoch time) with a microsecond resolution. In every flight folder, a YAML metadata file summarizes the camera and light setting, along with the type of track. The total time of flight and meters traveled are also included. The folder and file structure of the dataset are shown in Figure E.5



```
https://github.com/tii-racing/drone-racing-dataset/data/
└ [autonomous|piloted]
   └ flight-[01-12][a|p]-[ellipse|lemniscate|trackRATM]/
      ├ csv_raw/
      │  ├ ros2bag_dump/
      │  │  ├ battery_flight-01a-ellipse.csv
      │  │  ├ channels_flight-01a-ellipse.csv
      │  │  ├ *ctbr_flight-01a-ellipse.csv
      │  │  ├ drone_state_flight-01a-ellipse.csv
      │  │  ├ imu_flight-01a-ellipse.csv
      │  │  ├ motors_thrust_flight-01a-ellipse.csv
      │  │  └ *reference_flight-01a-ellipse.csv
      │  ├ camera_flight-01a-ellipse.csv
      │  ├ gate_corners_flight-01a-ellipse.csv
      │  └ mocap_flight-01a-ellipse.csv
      ├ ros2bag_flight-01a-ellipse/
      │  ├ metadata.yaml
      │  └ ros2bag_flight-01a-ellipse.db3
      ├ camera_flight-01a-ellipse.zip
      ├ label_flight-01a-ellipse.zip
      ├ flight-01a-ellipse_500hz_freq_sync.csv
      ├ flight-01a-ellipse_cam_ts_sync.csv
      └ metadata_flight-01a-ellipse.yaml
```

* Only in the autonomous flight folders.

Figure E.5: Folder and file structure of the dataset.

E.5.1 Camera-aligned and Uniform-sampling CSVs

For each flight, we pre-compiled two easy-to-use, comprehensive CSV files that include all the data detailed in the subsequent sections, aligned via interpolation. In each `[FLIGHT]_cam_ts_sync.csv` file, we use the timestamps from the camera frames, and all other data points are linearly interpolated to align with these timestamps. Conversely, in each `[FLIGHT]_500hz_freq_sync.csv` file, timestamps are sampled at a uniform 500Hz frequency between the first and last camera timestamps. All numerical data are again interpolated and aligned with timestamps, and each row references the file name of the camera frame with the closest timestamp. All the columns in each of these CSVs are presented in Table E.4.

Table E.4: Data available in the precompiled `cam_ts_sync.csv` and `500Hz_freq_sync.csv` (Sec. E.5.1)

Column Number and Quantity Name	Unit	Data Type
0. `elapsed_time`	s	float
1. `timestamp`	μs	int
2. `img_filename`	n/a	string
3. `accel_[x|y|z]`	m/s^2	float
6. `gyro_[x|y|z]`	rad/s	float
9. `thrust[0-3]`	1	float $\in [0,1]$
13. `channels_[roll|pitch|thrust|yaw]`	1	int $\in [1000,2000]$
17. `aux[1-4]`	1	int $\in [1000,2000]$
21. `vbat`	V	float
22. `drone_[x|y|z]`	m	float
25. `drone_[roll|pitch|yaw]`	rad	float
28. `drone_velocity_linear_[x|y|z]`	m/s	float
31. `drone_velocity_angular_[x|y|z]`	rad/s	float
34. `drone_residual`	m	float
35. `drone_rot[[0-8]]`	1	float
44. `gate[1-7]_int_[x|y|z]`	m	float
56. `gate[1-7]_int_[roll|pitch|yaw]`	rad	float
68. `gate[1-7]_int_residual`	m	float
72. `gate[1-7]_int_rot[[0-8]]`	1	float
108. `gate[1-7]_marker[1-4]_[x|y|z]`	m	float

E.5.2 Drone and Gates' Poses from Motion Capture

For each flight, files `gate_corners_[FLIGHT].csv` contain the timestamped x, y, and z coordinates of all the gates' markers, in meters. Files `mocap_[FLIGHT].csv` contain the poses of all the rigid bodies, i.e., the drone and the gates. Each pose comprises x, y, z, roll, pitch, yaw, measurement residual, and the orientation described by a 3×3 column-major order matrix. Poses are in meters and radiants.

E.5.3 Camera Frames

The images from the Arducam IMX219 Bayer camera are recorded at 120 FPS with a resolution of 640×480px and saved as JPEG files. They are provided as a ZIP file along with a `camera_[FLIGHT].csv` which contains the timestamps of the acquisition of the frame and the name of the corresponding JPEG file.

E.5.4 Image Labels

For each image, the gates' bounding boxes and internal corner labels are given as a TXT file with the same name as the JPEG file. Each line in a TXT file represents a single gate in the form:

$$0\ c_x\ c_y\ w\ h\ tl_x\ tl_y\ tl_v\ tr_x\ tr_y\ tr_v\ br_x\ br_y\ br_v\ bl_x\ bl_y\ bl_v$$

where 0 is the class label for a gate (the only class in our dataset); $c_x, c_y, w, h \in [0,1]$ are its bounding box center's coordinates, width, and height, respectively; and $tl_x, tl_y \in [0,1]$, $tl_v \in [0;2]$ are the coordinates and visibility (0 outside the image boundaries; 2 inside the image boundaries) of the top-left internal corner. Similarly for tr, bl, br, the top-right, bottom-left, and bottom-right corners. All values are in pixel coordinates normalized with respect to image size. The keypoints label format follows the COCO definition [207]. The labels are provided as a ZIP file in `label_[FLIGHT].zip`.

E.5.5 On-board Data from the Quadrotor Electronics

From the FCU in Subsubsection E.3.1, we record the following measurements: battery's voltage (Volts), at 50Hz (`battery_[FLIGHT].csv`); IMU, i.e., accelerometer (m/s^2) and gyroscope (rad/s) x, y, and z axes in the East-North-Up (ENU) board frame, at 500Hz (`imu_[FLIGHT].csv`); single motor thrust feedback, normalized between 0 and 1, for all four motors, at 100Hz (`motors_thrust_[FLIGHT].csv`); RC's channels, i.e. roll, pitch, thrust, yaw, aux1, aux2, aux3, and aux4 values of the sticks between 1000 and 2000, at 100Hz (`channels_[FLIGHT].csv`, only for human-piloted flights).

E.5.6 On-board Data from the Autonomous Module

From the NVIDIA Orin in Subsubsection E.3.1, we record the following measurements: the drone state, i.e., position (m), orientation (quaternion), velocity (m/s), and angular velocity (rad/s), at 275Hz, sent as ground truth from the MoCap system to the drone, with the time delay of the communication channel (`drone_state_[FLIGHT].csv`).

Furthermore, only for the autonomous flights, we also record: the controller's reference, i.e., position (m), orientation (quaternion), linear (m/s) and angular velocity (rad/s), acceleration (m/s^2), jerk (m/s^3), heading (rad), and heading rate (rad/s), computed at 100Hz for the ellipse and lemniscate, and 500Hz for the race track (`reference_[FLIGHT].csv`); the collective thrust and body rates (CTBR) computed by the autonomous controller, i.e., the normalized thrust (N/kg i.e. m/s^2), roll, pitch, and yaw rates (rad/s), at 275Hz for the PD controller, and 160Hz for the MPC, (`ctbr_[FLIGHT].csv`); and the RC's channels sent to the FCU calculated from the CTBR commands, i.e., roll, pitch, thrust, yaw, aux1, aux2, aux3, and aux4 values of the sticks, at 275Hz and 160Hz for PD and MPC respectively, (`channels_[FLIGHT].csv`).

E.6 Visualization and Post-processing Scripts

Along with the dataset, we provide an installable[4] open-source Python repository comprising of a set of processing scripts to visualize and manipulate the data.

`data_interpolation.py` can be used to re-sample and align the data at an arbitrary frequency using linear interpolation. Its output is a CSV file containing the aligned, re-sampled data from all the sources in Section E.5. First, one selects a flight by passing its id, e.g., `flight-01a-ellipse` as an argument. Then, one can choose which synchronization option to use. In option *1*, one must choose a new frequency for which timestamps are generated, using the first and last camera timestamp as a window. All the data are then interpolated to the generated timestamps. For the camera frames, the one with the closest timestamp is referred to by file name. In option *2*, the timestamps of the camera CSV are used as a reference instead.

```
$ python3 data_interpolation.py \
> --flight flight-01a-ellipse \
> --sync-option 1 --freq 200
Loading and pre-processing CSVs...
Using frequency 200 to interpolate all data
Syncing dataframes...
Sync complete.
Saving final CSV...
Final CSV saved.
```

`data_plotting.py` is a script to simultaneously visualize multiple sensor data using customizable subplots. One can select which subplots to include by means of command-line arguments. The script facilitates quick insights into the drone's flight behavior and performance, such as its 3D trajectory and the gate positions, as well as pose, velocities, accelerations, battery voltage, RC channels, etc.; an example of the script output is shown in Figure E.6.

```
$ python3 data_plotting.py \
> --csv-file flight-01a-ellipse_cam_ts_sync.csv \
> --subplots 3d
```

`label_visualization.py` is the script to visualize the label annotations on the images. It allows one to skim the images of a flight, read the associated YOLO-style label file, and plot the gates' bounding boxes and internal corner keypoints. An individual frame example is shown in Figure E.7.

```
$ python3 label_visualization.py \
> --flight flight-01a-ellipse
```

`create_std_bag.py` is a utility script that reads all the data from a user-specified flight and creates a new ROS2 bag with the same data (including images) but only using standard messages from the ROS2 `msgs` library. All the messages are timestamped in nanoseconds. It may take several GBs.

```
python3 create_std_bag.py \
> --flight flight-01a-ellipse
```

[4] `github.com/tii-racing/drone-racing-dataset/blob/ main/README.md`

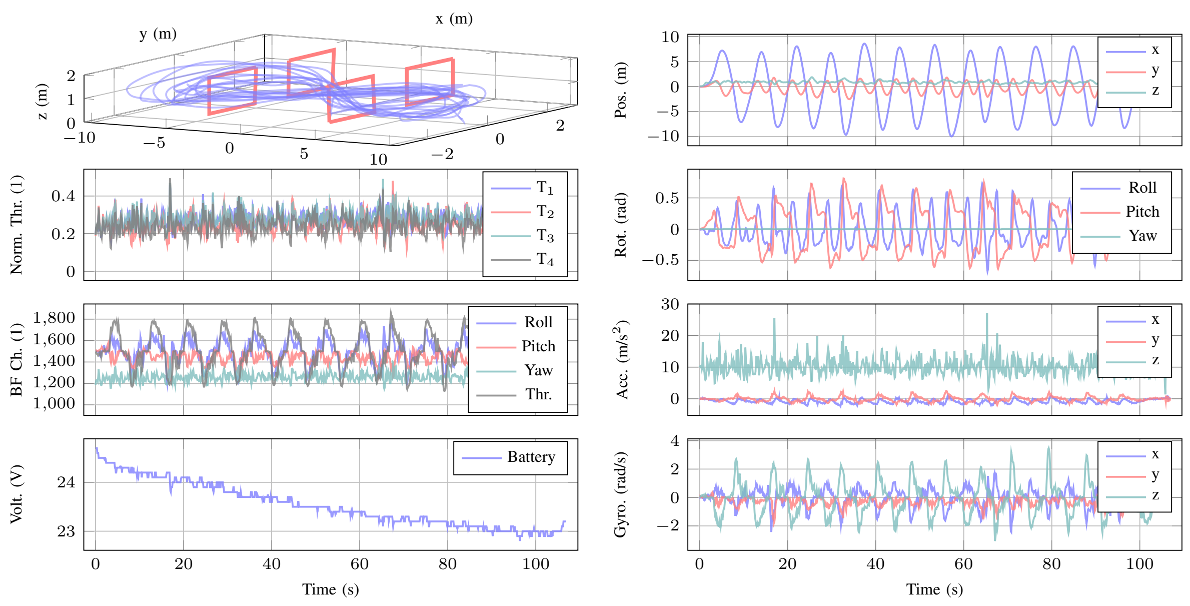


Figure E.6: Sample output of the utility script `data_plotting.py` (Section E.6), plotting 21 different data points for `flight-07p-lemniscate`, a piloted, 13-lap flight on the lemniscate trajectory.

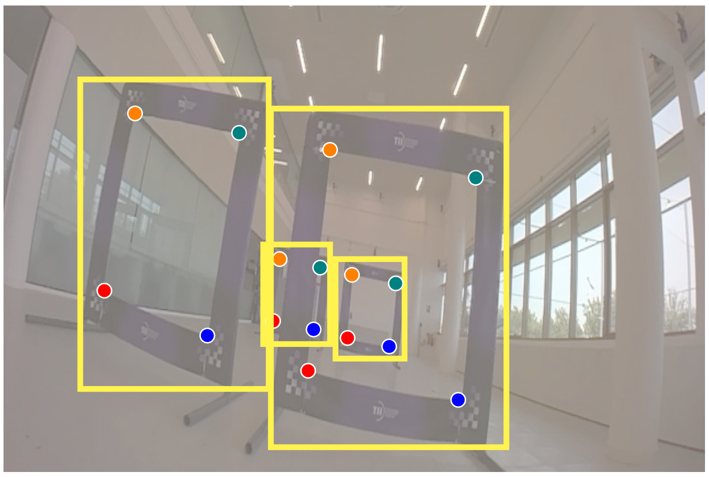


Figure E.7: Sample output of script `label_visualization.py` from Section E.6, overlaying the bounding boxes and keypoints (top-left, top-right, bottom-right, and bottom-left corners) for all obstructed and unobstructed gates in one of the frames from `flight-02a-ellipse`.

E.7 Conclusions

The development of autonomous racing drones requires simultaneously tackling challenging perception, state estimation, and control tasks—in real time—under limited computational resources. With this dataset, we created a one-stop resource to develop and evaluate new algorithms for autonomous drone racing. Our work is comprehensive of both aggressive autonomous and piloted flight; high-resolution, high-frequency visual, inertial, and motion capture data; commands and control inputs; multiple light settings; and corner-level labeling of drone racing gates. Along with the data, we open-sourced the scripts used to parse and visualize them. To further democratize autonomous drone racing research, we also released the parts list and instructions to recreate our flight platform, using commercial off-the-shelf components, hoping to see it re-created and used by researchers around the world.

Bibliography

[1] K. L. Wong, M. Bosello, R. Tse, C. Falcomer, C. Rossi, and G. Pau, “Li-ion batteries state-of-charge estimation using deep lstm at various battery specifications and discharge cycles,” in *Proceedings of the Conference on Information Technology for Social Good*, ser. GoodIT ’21, ACM, 2021, pp. 85–90. DOI: 10.1145/3462203.3475878.

[2] M. Bosello, C. Falcomer, C. Rossi, and G. Pau, “To charge or to sell? ev pack useful life estimation via lstms, cnns, and autoencoders,” *Energies*, vol. 16, no. 6, 2023. DOI: 10.3390/en16062837.

[3] M. Bosello, R. Tse, and G. Pau, “Train in austria, race in montecarlo: Generalized rl for cross-track f1tenth lidar-based races,” in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2022, pp. 290–298. DOI: 10.1109/CCNC49033.2022.9700730.

[4] Y. Chen, R. Tse, M. Bosello, D. Aguiari, S.-K. Tang, and G. Pau, “Enabling deep reinforcement learning autonomous driving by 3D-LiDAR point clouds,” in *Fourteenth International Conference on Digital Image Processing (ICDIP 2022)*, vol. 12342, SPIE, 2022. DOI: 10.1117/12.2644369.

[5] M. Bosello, D. Aguiari, Y. Keuter, *et al.*, “Race against the machine: A fully-annotated, open-design dataset of autonomous and piloted high-speed flight,” *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3799–3806, 2024. DOI: 10.1109/LRA.2024.3371288.

[6] Y. Wu, Q. Xue, J. Shen, Z. Lei, Z. Chen, and Y. Liu, “State of health estimation for lithium-ion batteries based on healthy features and long short-term memory,” *IEEE Access*, vol. 8, pp. 28 533–28 547, 2020. DOI: 10.1109/ACCESS.2020.2972344.

[7] N. Johnson, “19 - battery technology for co2 reduction,” in *Alternative Fuels and Advanced Vehicle Technologies for Improved Environmental Performance*, Woodhead Publishing, 2014, pp. 582–631. DOI: 10.1533/9780857097422.3.582.

[8] M. Hannan, M. Lipu, A. Hussain, and A. Mohamed, “A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: Challenges and recommendations,” *Renewable and Sustainable Energy Reviews*, vol. 78, pp. 834–854, 2017. DOI: 10.1016/j.rser.2017.05.001.

[9] D. Hanover, A. Loquercio, L. Bauersfeld, *et al.*, “Autonomous drone racing: A survey,” *IEEE Transactions on Robotics*, vol. 40, pp. 3044–3067, 2024. DOI: 10.1109/TRO.2024.3400838.

[10] J. Betz, H. Zheng, A. Liniger, *et al.*, “Autonomous vehicles on the edge: A survey on autonomous vehicle racing,” *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022. DOI: 10.1109/OJITS.2022.3181510.

[11] T. Herrmann, F. Sauerbeck, M. Bayerlein, J. Betz, and M. Lienkamp, “Optimization-based real-time-capable energy strategy for autonomous electric race cars,” *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12-05-01-0005, pp. 45–59, 2022. DOI: 10.4271/12-05-01-0005.

[12] L. Bauersfeld and D. Scaramuzza, “Range, endurance, and optimal speed estimates for multicopters,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2953–2960, 2022. DOI: 10.1109/LRA.2022.3145063.

[13] C. Vidal, P. Malysz, P. Kollmeyer, and A. Emadi, “Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art,” *IEEE Access*, vol. 8, pp. 52 796–52 814, 2020. DOI: 10.1109/ACCESS.2020.2980961.

[14] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, “Champion-level drone racing using deep reinforcement learning,” *Nature*, vol. 620, no. 7976, pp. 982–987, 2023. DOI: 10.1038/s41586-023-06419-4.

[15] N. Williard, W. He, C. Hendricks, and M. Pecht, “Lessons learned from the 787 dreamliner issue on lithium-ion battery reliability,” *Energies*, vol. 6, no. 9, pp. 4682–4695, 2013. DOI: 10.3390/en6094682.

[16] NASA, *Mars global surveyor (mgs) spacecraft loss of contact*, https://www.nasa.gov/pdf/174244main_mgs_white_paper_20070413.pdf, Apr. 2007.

[17] IEA, *Batteries and secure energy transitions*, https://www.iea.org/reports/batteries-and-secure-energy-transitions, IEA, 2024.

[18] A. Opitz, P. Badami, L. Shen, K. Vignarooban, and A. Kannan, “Can li-ion batteries be the panacea for automotive applications?” *Renewable and Sustainable Energy Reviews*, vol. 68, pp. 685–692, 2017. DOI: 10.1016/j.rser.2016.10.019.

[19] S. Manzetti and F. Mariasiu, “Electric vehicle battery technologies: From present state to future systems,” *Renewable and Sustainable Energy Reviews*, vol. 51, pp. 1004–1012, 2015. DOI: 10.1016/j.rser.2015.07.010.

[20] L. Song, K. Zhang, T. Liang, X. Han, and Y. Zhang, “Intelligent state of health estimation for lithium-ion battery pack based on big data analysis,” *Journal of Energy Storage*, vol. 32, p. 101 836, 2020. DOI: 10.1016/j.est.2020.101836.

[21] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications," *Journal of Power Sources*, vol. 241, pp. 680–689, 2013. DOI: 10.1016/j.jpowsour.2013.05.040.

[22] F. Karimi Pour, D. Theilliol, V. Puig, and G. Cembrano, "Health-aware control design based on remaining useful life estimation for autonomous racing vehicle," *ISA Transactions*, vol. 113, pp. 196–209, 2021. DOI: 10.1016/j.isatra.2020.03.032.

[23] K. W. E. Cheng, B. P. Divakar, H. Wu, K. Ding, and H. F. Ho, "Battery-management system (bms) and soc development for electrical vehicles," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 1, pp. 76–88, 2011. DOI: 10.1109/TVT.2010.2089647.

[24] G. L. Plett, *Battery management systems, Volume II: Equivalent-circuit methods*. Artech House, 2015.

[25] J. P. Rivera-Barrera, N. Muñoz-Galeano, and H. O. Sarmiento-Maldonado, "Soc estimation for lithium-ion batteries: Review and future challenges," *Electronics*, vol. 6, no. 4, p. 102, 2017. DOI: 10.3390/electronics6040102.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[27] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, and A. Emadi, "Long short-term memory networks for accurate state-of-charge estimation of li-ion batteries," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6730–6739, 2018. DOI: 10.1109/TIE.2017.2787586.

[28] R. Xiong, L. Li, and J. Tian, "Towards a smarter battery management system: A critical review on battery state of health monitoring methods," *Journal of Power Sources*, vol. 405, pp. 18–29, 2018. DOI: 10.1016/j.jpowsour.2018.10.019.

[29] S. Shen, M. Sadoughi, X. Chen, M. Hong, and C. Hu, "A deep learning method for online capacity estimation of lithium-ion batteries," *Journal of Energy Storage*, vol. 25, p. 100 817, 2019. DOI: 10.1016/j.est.2019.100817.

[30] Y. Chen, Y. He, Z. Li, L. Chen, and C. Zhang, "Remaining useful life prediction and state of health diagnosis of lithium-ion battery based on second-order central difference particle filter," *IEEE Access*, vol. 8, pp. 37 305–37 313, 2020. DOI: 10.1109/ACCESS.2020.2974401.

[31] X. Li, X. Shu, J. Shen, R. Xiao, W. Yan, and Z. Chen, "An on-board remaining useful life estimation algorithm for lithium-ion batteries of electric vehicles," *Energies*, vol. 10, no. 5, 2017. DOI: 10.3390/en10050691.

[32] S. S. Ng, Y. Xing, and K. L. Tsui, "A naive bayes model for robust remaining useful life prediction of lithium-ion battery," *Applied Energy*, vol. 118, pp. 114–123, 2014. DOI: 10.1016/j.apenergy.2013.12.020.

[33] G. dos Reis, C. Strange, M. Yadav, and S. Li, "Lithium-ion battery data and where to find it," *Energy and AI*, p. 100 081, 2021. DOI: 10.1016/j.egyai.2021.100081.

[34] B. Saha and K. Goebel, *Battery data set*, https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/#battery, NASA Ames Prognostics Data Repository, 2007.

[35] C. Hametner, S. Jakubek, and W. Prochazka, "Data-driven design of a cascaded observer for battery state of health estimation," in *2016 IEEE International Conference on Sustainable Energy Technologies (ICSET)*, 2016, pp. 180–185. DOI: 10.1109/ICSET.2016.7811778.

[36] D. Zhou, Z. Li, J. Zhu, H. Zhang, and L. Hou, "State of health monitoring and remaining useful life prediction of lithium-ion batteries based on temporal convolutional network," *IEEE Access*, vol. 8, pp. 53 307–53 320, 2020. DOI: 10.1109/ACCESS.2020.2981261.

[37] Y. Zhang, R. Xiong, H. He, and Z. Liu, "A lstm-rnn method for the lithuim-ion battery remaining useful life prediction," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 2017, pp. 1–4. DOI: 10.1109/PHM.2017.8079316.

[38] W. Zhang, X. Li, and X. Li, "Deep learning-based prognostic approach for lithium-ion batteries with adaptive time-series prediction and on-line validation," *Measurement*, vol. 164, p. 108 052, 2020. DOI: 10.1016/j.measurement.2020.108052.

[39] P. Li, Z. Zhang, Q. Xiong, *et al.*, "State-of-health estimation and remaining useful life prediction for the lithium-ion battery based on a variant long short term memory neural network," *Journal of Power Sources*, vol. 459, p. 228 069, 2020. DOI: 10.1016/j.jpowsour.2020.228069.

[40] L. Ren, L. Zhao, S. Hong, S. Zhao, H. Wang, and L. Zhang, "Remaining useful life prediction for lithium-ion battery: A deep learning approach," *IEEE Access*, vol. 6, pp. 50 587–50 598, 2018. DOI: 10.1109/ACCESS.2018.2858856.

[41] L. Ren, J. Dong, X. Wang, Z. Meng, L. Zhao, and M. J. Deen, "A data-driven auto-cnn-lstm prediction model for lithium-ion battery remaining useful life," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3478–3487, 2021. DOI: 10.1109/TII.2020.3008223.

[42] B. R. Kiran, I. Sobh, V. Talpaert, *et al.*, *Deep reinforcement learning for autonomous driving: A survey*, 2020. arXiv: 2002.00444 [cs.LG].

[43] B. D. Evans, R. Trumpp, M. Caccamo, *et al.*, *Unifying f1tenth autonomous racing: Survey, methods and benchmarks*, 2024. arXiv: 2402.18558 [cs.RO].

[44] B. Huval, T. Wang, S. Tandon, *et al.*, *An empirical evaluation of deep learning on highway driving*, 2015. arXiv: 1504.01716 [cs.RO].

[45] F. Leon and M. Gavrilescu, "A review of tracking and trajectory prediction methods for autonomous driving," *Mathematics*, vol. 9, no. 6, p. 660, 2021. DOI: 10.3390/math9060660.

[46] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd. MIT Press, 2018.

[47] A. Kendall, J. Hawke, D. Janz, *et al.*, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8248–8254. DOI: 10.1109/ICRA.2019.8793742.

[48] M. Bosello, R. Tse, and G. Pau, "Robot drivers: Learning to drive by trial & error," in *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, IEEE, 2019, pp. 284–290. DOI: 10.1109/MSN48538.2019.00061.

[49] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6713–6719. DOI: 10.1109/ICRA.2019.8793887.

[50] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, "The blackbird dataset: A large-scale dataset for uav perception in aggressive flight," in *Proceedings of the 2018 International Symposium on Experimental Robotics*, Springer, 2020, pp. 130–139. DOI: 10.1007/978-3-030-33950-0_12.

[51] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Case study: Verifying the safety of an autonomous racing car with a neural network controller," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2020. DOI: 10.1145/3365365.3382216.

[52] I. E. Agency, *Global EV Outlook 2020*. OECD Publishing, 2020, p. 276. DOI: 10.1787/d394399e-en.

[53] R. Korthauer, *Lithium-ion batteries: basics and applications*. Springer, 2018. DOI: 10.1007/978-3-662-53071-9.

[54] S. Ma, M. Jiang, P. Tao, *et al.*, "Temperature effect and thermal impact in lithium-ion batteries: A review," *Progress in Natural Science: Materials International*, vol. 28, no. 6, pp. 653–666, 2018. DOI: 10.1016/j.pnsc.2018.11.002.

[55] D. Ouyang, M. Chen, J. Liu, R. Wei, J. Weng, and J. Wang, "Investigation of a commercial lithium-ion battery under overcharge/over-discharge failure conditions," *RSC Advances*, vol. 8, no. 58, pp. 33 414–33 424, 2018. DOI: 10.1039/C8RA05564E.

[56] Y. Xu, M. Hu, A. Zhou, *et al.*, "State of charge estimation for lithium-ion batteries based on adaptive dual kalman filter," *Applied Mathematical Modelling*, vol. 77, pp. 1255–1272, 2020. DOI: 10.1016/j.apm.2019.09.011.

[57] G. Pistoia, *Lithium-ion batteries: advances and applications*. Elsevier Science, 2014.

[58] G. L. Plett, “Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 2. modeling and identification,” *Journal of Power Sources*, vol. 134, no. 2, pp. 262–276, 2004. DOI: 10.1016/j.jpowsour.2004.02.032.

[59] D. N. T. How, M. A. Hannan, M. S. Hossain Lipu, and P. J. Ker, “State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review,” *IEEE Access*, vol. 7, pp. 136 116–136 136, 2019. DOI: 10.1109/ACCESS.2019.2942213.

[60] C. Campestrini, T. Heil, S. Kosch, and A. Jossen, “A comparative study and review of different kalman filters by applying an enhanced validation method,” *Journal of Energy Storage*, vol. 8, pp. 142–159, 2016. DOI: 10.1016/j.est.2016.10.004.

[61] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020. DOI: 10.1007/s10462-020-09825-6.

[62] W. De Mulder, S. Bethard, and M.-F. Moens, “A survey on the application of recurrent neural networks to statistical language modeling,” *Computer Speech and Language*, vol. 30, no. 1, pp. 61–98, 2015. DOI: 10.1016/j.csl.2014.09.005.

[63] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947.

[64] W. Yin, K. Kann, M. Yu, and H. Schütze, *Comparative study of cnn and rnn for natural language processing*, 2017. arXiv: 1702.01923 [cs.CL].

[65] P. Kollmeyer, *Panasonic 18650PF Li-ion Battery Data*, Mendeley Data, 2018. DOI: 10.17632/wykht8y7tg.1.

[66] F. Yang, X. Song, F. Xu, and K.-L. Tsui, “State-of-charge estimation of lithium-ion batteries via long short-term memory network,” *IEEE Access*, vol. 7, pp. 53 792–53 799, 2019. DOI: 10.1109/ACCESS.2019.2912803.

[67] X. Song, F. Yang, D. Wang, and K.-L. Tsui, “Combined cnn-lstm network for state-of-charge estimation of lithium-ion batteries,” *IEEE Access*, vol. 7, pp. 88 894–88 902, 2019. DOI: 10.1109/ACCESS.2019.2926517.

[68] S. Cui, X. Yong, S. Kim, S. Hong, and I. Joe, “An lstm-based encoder-decoder model for state-of-charge estimation of lithium-ion batteries,” in *Intelligent Algorithms in Software Engineering*, Springer, 2020, pp. 178–188. DOI: 10.1007/978-3-030-51965-0_15.

[69] P. Kollmeyer, C. Vidal, M. Naguib, and M. Skells, *LG 18650HG2 Li-ion Battery Data and Example Deep Neural Network xEV SOC Estimator Script*, 2020. DOI: 10.17632/cp3473x7xv.3.

[70] United States Environmental Protection Agency, *EPA Urban Dynamometer Driving Schedule (UDDS)*, https://www.epa.gov/emission-standards-reference-guide/epa-urban-dynamometer-driving-schedule-udds, 2020.

[71] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Curran Associates Inc., 2017, pp. 972–981.

[72] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

[73] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[74] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, 1964. DOI: 10.1214/aoms/1177703732.

[75] J. Hofmann, D. Guan, K. Chalvatzis, and H. Huo, "Assessment of electrical vehicles as a successful driver for reducing co2 emissions in china," *Applied Energy*, vol. 184, pp. 995–1003, 2016. DOI: 10.1016/j.apenergy.2016.06.042.

[76] Y. Zou, S. Wei, F. Sun, X. Hu, and Y. Shiao, "Large-scale deployment of electric taxis in beijing: A real-world analysis," *Energy*, vol. 100, pp. 25–39, 2016. DOI: 10.1016/j.energy.2016.01.062.

[77] US Environmental Protection Agency (EPA), *Global Greenhouse Gas Emissions Data*, https://www.epa.gov/ghgemissions/global-greenhouse-gas-emissions-data, 2014.

[78] European Commission, *Air pollution from the main sources - Air emissions from road vehicles*, https://ec.europa.eu/environment/air/sources/road.htm, 2012.

[79] M. J. Kleeman, J. J. Schauer, and G. R. Cass, "Size and composition distribution of fine particulate matter emitted from motor vehicles," *Environmental Science & Technology*, vol. 34, no. 7, pp. 1132–1142, 2000. DOI: 10.1021/es981276y.

[80] I. Kheirbek, J. Haney, S. Douglas, K. Ito, and T. Matte, "The contribution of motor vehicle emissions to ambient fine particulate matter public health impacts in new york city: A health burden assessment," *Environmental Health*, vol. 15, no. 1, pp. 1–14, 2016. DOI: 10.1186/s12940-016-0172-6.

[81] C. D. Koolen and G. Rothenberg, "Air pollution in europe," *ChemSusChem*, vol. 12, no. 1, pp. 164–172, 2019. DOI: https://doi.org/10.1002/cssc.201802292.

[82] H. R. Anderson, R. W. Atkinson, J. L. Peacock, M. J. Sweeting, and L. Marston, "Ambient particulate matter and health effects: Publication bias in studies of short-term associations," *Epidemiology*, pp. 155–163, 2005. DOI: 10.1097/01.ede.0000152528.22746.0f.

[83] B. Brunekreef and B. Forsberg, "Epidemiological evidence of effects of coarse airborne particles on health," *European Respiratory Journal*, vol. 26, no. 2, pp. 309–318, 2005. DOI: 10.1183/09031936.05.00001805.

[84] S. Zhang, B. Zhai, X. Guo, K. Wang, N. Peng, and X. Zhang, "Synchronous estimation of state of health and remaining useful lifetime for lithium-ion battery using the incremental capacity and artificial neural networks," *Journal of Energy Storage*, vol. 26, p. 100 951, 2019. DOI: 10.1016/j.est.2019.100951.

[85] D. Liu, J. Zhou, and Y. Peng, "Data-driven prognostics and remaining useful life estimation for lithium-ion battery: A review," *Instrumentation*, vol. 1, 2014.

[86] X. Hu, D. Cao, and B. Egardt, "Condition monitoring in advanced battery management systems: Moving horizon estimation using a reduced electrochemical model," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 167–178, 2017. DOI: 10.1109/TMECH.2017.2675920.

[87] C. I. Valdes-Dapena and Peter, *Hyundai's recall of 82,000 electric cars is one of the most expensive in history*, https://edition.cnn.com/2021/02/25/tech/hyundai-ev-recall/index.html, May 2021.

[88] A. J. Hawkins, *GM recalls 68,000 electric Chevy Bolts over battery fire concerns*, https://www.theverge.com/2020/11/13/21564217/gm-chevy-bolt-recall-battery-fire-lg-chem, Nov. 2020.

[89] B. Bilgin, P. Magne, P. Malysz, *et al.*, "Making the case for electrified transportation," *IEEE Transactions on Transportation Electrification*, vol. 1, no. 1, pp. 4–17, 2015. DOI: 10.1109/TTE.2015.2437338.

[90] A. Mahmoudzadeh Andwari, A. Pesiridis, S. Rajoo, R. Martinez-Botas, and V. Esfahanian, "A review of battery electric vehicle technology and readiness levels," *Renewable and Sustainable Energy Reviews*, vol. 78, pp. 414–430, 2017. DOI: 10.1016/j.rser.2017.03.138.

[91] L. Lu, X. Han, J. Li, J. Hua, and M. Ouyang, "A review on the key issues for lithium-ion battery management in electric vehicles," *Journal of Power Sources*, vol. 226, pp. 272–288, 2013. DOI: 10.1016/j.jpowsour.2012.10.060.

[92] Y. Fan, F. Xiao, C. Li, G. Yang, and X. Tang, "A novel deep learning framework for state of health estimation of lithium-ion battery," *Journal of Energy Storage*, vol. 32, p. 101 741, 2020. DOI: 10.1016/j.est.2020.101741.

[93] International Electrotechnical Commission, *Secondary lithium-ion cells for the propulsion of electric road vehicles—part 2: Reliability and abuse testing*, IEC-62660-2, 2018.

[94] ISO, *Iso electrically propelled road vehicles—test specification for lithium-ion traction battery packs and systems—part 3: Safety performance requirements*, ISO 12405-3, 2018.

[95] IEEE, *Ieee recommended practice for maintenance, testing, and replacement of vented lead-acid batteries for stationary applications*, IEEE Std 450-2020 (Revision IEEE Std 450-2010), 2021, pp. 1–71.

[96] T. Q. Duong, "Usabc and pngv test procedures," *Journal of Power Sources*, vol. 89, no. 2, pp. 244–248, 2000. DOI: 10.1016/S0378-7753(00)00439-0.

[97] Y. Xing, E. W. Ma, K.-L. Tsui, and M. Pecht, "An ensemble model for predicting the remaining useful performance of lithium-ion batteries," *Microelectronics Reliability*, vol. 53, no. 6, pp. 811–820, 2013. DOI: 10.1016/j.microrel.2012.12.003.

[98] X. Hu, F. Feng, K. Liu, L. Zhang, J. Xie, and B. Liu, "State estimation for advanced battery management: Key challenges and future trends," *Renewable and Sustainable Energy Reviews*, vol. 114, p. 109 334, 2019. DOI: 10.1016/j.rser.2019.109334.

[99] P. Venugopal and V. T., "State-of-health estimation of li-ion batteries in electric vehicle using indrnn under variable load condition," *Energies*, vol. 12, no. 22, 2019. DOI: 10.3390/en12224338.

[100] L. Ahmadi, A. Yip, M. Fowler, S. B. Young, and R. A. Fraser, "Environmental feasibility of re-use of electric vehicle batteries," *Sustainable Energy Technologies and Assessments*, vol. 6, pp. 64–74, 2014. DOI: 10.1016/j.seta.2014.01.006.

[101] G. Harper, R. Sommerville, E. Kendrick, *et al.*, "Recycling lithium-ion batteries from electric vehicles," *Nature*, vol. 575, no. 7781, pp. 75–86, 2019. DOI: 10.1038/s41586-019-1682-5.

[102] A. Pražanová, V. Knap, and D.-I. Stroe, "Literature review, recycling of lithium-ion batteries from electric vehicles, part ii: Environmental and economic perspective," *Energies*, vol. 15, no. 19, 2022. DOI: 10.3390/en15197356.

[103] M. H. Lipu, M. Hannan, A. Hussain, *et al.*, "A review of state of health and remaining useful life estimation methods for lithium-ion battery in electric vehicles: Challenges and recommendations," *Journal of Cleaner Production*, vol. 205, pp. 115–133, 2018. DOI: 10.1016/j.jclepro.2018.09.065.

[104] H. Dai, G. Zhao, M. Lin, J. Wu, and G. Zheng, "A novel estimation method for the state of health of lithium-ion battery using prior knowledge-based neural network and markov chain," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 10, pp. 7706–7716, 2019. DOI: 10.1109/TIE.2018.2880703.

[105] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, pp. 1097–1105.

[106] B. Bole, C. Kulkarni, and M. Daigle, *Randomized battery usage data set*, http://ti.arc.nasa.gov/project/prognostic-data-repository, NASA Ames Prognostics Data Repository, 2014.

[107] L. Ungurean, M. V. Micea, and G. Cârstoiu, "Online state of health prediction method for lithium-ion batteries, based on gated recurrent unit neural networks," *International Journal of Energy Research*, vol. 44, no. 8, pp. 6767–6777, 2020. DOI: 10.1002/er.5413.

[108] M. Maggipinto, C. Masiero, A. Beghi, and G. A. Susto, "A convolutional autoencoder approach for feature extraction in virtual metrology," *Procedia Manufacturing*, vol. 17, pp. 126–133, 2018. DOI: 10.1016/j.promfg.2018.10.023.

[109] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[110] Y. Tang, Q. Zhang, Y. Li, G. Wang, and Y. Li, "Recycling mechanisms and policy suggestions for spent electric vehicles' power battery -a case of beijing," *Journal of Cleaner Production*, vol. 186, pp. 388–406, 2018. DOI: 10.1016/j.jclepro.2018.03.043.

[111] H. Hao, W. Xu, F. Wei, C. Wu, and Z. Xu, "Reward–penalty vs. deposit–refund: Government incentive mechanisms for ev battery recycling," *Energies*, vol. 15, no. 19, 2022. DOI: 10.3390/en15196885.

[112] E. Ackerman, "Self-driving cars were just around the corner—in 1960," *IEEE Spectrum*, 2016. eprint: https://spectrum.ieee.org/selfdriving-cars-were-just-around-the-cornerin-1960.

[113] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA Grand Challenge: The Great Robot Race*. Springer, 2007. DOI: 10.1007/978-3-540-73429-1.

[114] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, 2009. DOI: 10.1007/978-3-642-03991-1.

[115] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, 1989. DOI: 10.1162/neco.1989.1.4.541.

[116] V. Talpaert, I. Sobh, B. R. Kiran, *et al.*, *Exploring applications of deep reinforcement learning for real-world autonomous driving systems*, 2019. arXiv: 1901.01536 [cs.LG].

[117] CBINSIGHT, *40+ corporations working on autonomous vehicles*, https://www.cbinsights.com/research.

[118] C. K. Toth, Z. Koppanyi, and M. G. Lenzano, "New source of geospatial data: Crowdsensing by assisted and autonomous vehicle technologies," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-4/W8, 2018. DOI: 10.5194/isprs-archives-XLII-4-W8-211-2018.

[119] H. G. Seif and X. Hu, “Autonomous driving in the icity—hd maps as a key challenge of the automotive industry,” *Engineering*, vol. 2, no. 2, 2016. DOI: 10.1016/J.ENG.2016.02.010.

[120] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013. DOI: 10.1177/0278364913495721.

[121] H. Zhu, J. Yu, A. Gupta, *et al.*, “The ingredients of real world robotic reinforcement learning,” in *International Conference on Learning Representations*, 2020.

[122] M. O’Kelly, V. Sukhil, H. Abbas, *et al.*, *F1/10: An open-source autonomous cyber-physical platform*, 2019. arXiv: 1901.08567 [cs.RO].

[123] M. O’Kelly, H. Zheng, D. Karthik, and R. Mangharam, “F1tenth: An open-source evaluation environment for continuous control and reinforcement learning,” *Proceedings of Machine Learning Research*, vol. 123, 2020.

[124] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, *Playing atari with deep reinforcement learning*, 2013. arXiv: 1312.5602 [cs.LG].

[125] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep reinforcement learning framework for autonomous driving,” *Electronic Imaging*, vol. 29, no. 19, p. 70, 2017. DOI: 10.2352/ISSN.2470-1173.2017.19.AVM-023.

[126] D. Li, D. Zhao, Q. Zhang, and Y. Chen, “Reinforcement learning and deep learning based lateral control for autonomous driving [application notes],” *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, 2019. DOI: 10.1109/MCI.2019.2901089.

[127] A. Yu, R. Palefsky-Smith, and R. Bedi, “Deep reinforcement learning for simulated autonomous vehicle control,” Stanford University, StuDocu, 2016.

[128] Y. You, X. Pan, Z. Wang, and C. Lu, “Virtual to real reinforcement learning for autonomous driving,” *CoRR*, 2017. arXiv: 1704.03952.

[129] J. Shan and C. Toth, *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, 2009. DOI: https://doi.org/10.1201/9781420051438.

[130] National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, *Lidar 101: An introduction to lidar technology, data, and applications.* 2012.

[131] Y. Li, L. Ma, Z. Zhong, *et al.*, “Deep learning for lidar point clouds in autonomous driving: A review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3412–3432, 2021. DOI: 10.1109/TNNLS.2020.3015992.

[132] A. Brunnbauer, L. Berducci, A. Brandstätter, *et al.*, *Model-based versus model-free deep reinforcement learning for autonomous racing cars*, 2021. arXiv: 2103.04909 [cs.LG].

[133] A. Olsson and F. Rosberg, “Domain transfer for end-to-end reinforcement learning,” M.S. thesis, Halmstad University, 2020.

[134] S. Chen, M. Wang, W. Song, Y. Yang, Y. Li, and M. Fu, “Stabilization approaches for reinforcement learning-based end-to-end autonomous driving,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, 2020. DOI: 10.1109/TVT.2020.2979493.

[135] S. Hossain, O. Doukhi, Y. Jo, and D.-J. Lee, “Deep reinforcement learning-based ros-controlled rc car for autonomous path exploration in the unknown environment,” in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 1231–1236. DOI: 10.23919/ICCAS50221.2020.9268370.

[136] M. Bosello, “Integrating bdi and reinforcement learning: The case study of autonomous driving,” M.S. thesis, Università di Bologna, 2020. DOI: 10.13140/RG.2.2.13072.23044.

[137] *Jetson TX2 Module*, https://developer.nvidia.com/embedded/jetson-tx2.

[138] *Orbitty carrier for nvidia*, https://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1.

[139] Hokuyo Automatic Corporation, *Scanning laser range finder smart-urg mini ust-10lx specification*, https://hokuyo-usa.com/application/files/3515/8947/8336/UST-10LX_Specifications.pdf, 2016.

[140] M. Quigley, K. Conley, B. Gerkey, *et al.*, “Ros: An open-source robot operating system,” in *ICRA Workshop on Open Source Software*, vol. 3, 2009.

[141] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, *Concrete problems in ai safety*, 2016. arXiv: 1606.06565 [cs.AI].

[142] N. M. Nawi, W. H. Atomi, and M. Rehman, “The effect of data pre-processing on optimized training of artificial neural networks,” *Procedia Technology*, vol. 11, pp. 32–39, 2013, 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013. DOI: https://doi.org/10.1016/j.protcy.2013.12.159.

[143] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. DOI: 10.1038/nature14236.

[144] J. Gu, Z. Wang, J. Kuen, *et al.*, “Recent advances in convolutional neural networks,” *Pattern Recognition*, vol. 77, pp. 354–377, 2018. DOI: https://doi.org/10.1016/j.patcog.2017.10.013.

[145] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable slam system with full 3d motion estimation,” in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160. DOI: 10.1109/SSRR.2011.6106777.

[146] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni, “Race driver model,” *Computers & Structures*, vol. 86, no. 13, pp. 1503–1516, 2008. DOI: 10.1016/j.compstruc.2007.04.028.

[147] V. Mnih, A. P. Badia, M. Mirza, *et al.*, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of Machine Learning Research*, vol. 48, PMLR, 2016.

[148] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks*, 2017. arXiv: 1703.03400 [cs.LG].

[149] M. Kamp, L. Adilova, J. Sicking, *et al.*, *Efficient decentralized deep learning by dynamic model averaging*, 2018. arXiv: 1807.03210 [cs.LG].

[150] M. Pfeiffer, S. Shukla, M. Turchetta, *et al.*, “Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, 2018. DOI: 10.1109/LRA.2018.2869644.

[151] National Highway Traffic Safety Administration (NHTSA), *Automated vehicles for safety*, https://www.nhtsa.gov/technology-innovation\/automated-vehicles-safety.

[152] J. Levinson, J. Askeland, J. Becker, *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 163–168. DOI: 10.1109/IVS.2011.5940562.

[153] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2009.

[154] H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, “Pedestrian recognition and tracking using 3d lidar for autonomous vehicle,” *Robotics and Autonomous Systems*, vol. 88, pp. 71–78, 2017. DOI: 10.1016/j.robot.2016.11.014.

[155] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer, “Multi-target tracking using a 3d-lidar sensor for autonomous vehicles,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 881–886. DOI: 10.1109/ITSC.2013.6728343.

[156] M. Weinmann, *Reconstruction and Analysis of 3D Scenes, From Irregularly Distributed 3D Points to Object Classes*. Springer, 2016. DOI: 10.1007/978-3-319-29246-5.

[157] B. Yang, J. Wang, R. Clark, *et al.*, “Learning object bounding boxes for 3d instance segmentation on point clouds,” in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[158] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5MB model size*, 2016. arXiv: 1602.07360 [cs.CV].

[159] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. DOI: 10.1126/science.aar6404.

[160] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[161] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette, "End-to-end driving in a realistic racing game with deep reinforcement learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 474–475. DOI: 10.1109/CVPRW.2017.64.

[162] M. Bansal, A. Krizhevsky, and A. Ogale, *Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst*, 2018. arXiv: 1812.03079 [cs.RO].

[163] Z. Xu, C. Tang, and M. Tomizuka, "Zero-shot deep reinforcement learning driving policy transfer for autonomous vehicles based on robust control," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2865–2871. DOI: 10.1109/ITSC.2018.8569612.

[164] X. Wang, Q. Huang, A. Celikyilmaz, *et al.*, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6622–6631. DOI: 10.1109/CVPR.2019.00679.

[165] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, *Learning navigation behaviors end-to-end with autorl*, 2019. arXiv: 1809.10124 [cs.RO].

[166] A. Faust, O. Ramirez, M. Fiser, *et al.*, *Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning*, 2018. arXiv: 1710.03937 [cs.AI].

[167] A. Francis, A. Faust, H.-T. L. Chiang, *et al.*, *Long-range indoor navigation with prm-rl*, 2020. arXiv: 1902.09458 [cs.RO].

[168] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5648–5656. DOI: 10.1109/CVPR.2016.609.

[169] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 945–953. DOI: 10.1109/ICCV.2015.114.

[170] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[171] Z. Wu, S. Song, A. Khosla, *et al.*, " 3D ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801.

[172] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

[173] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[174] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87, PMLR, 2018, pp. 237–252.

[175] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, *Driving policy transfer via modularity and abstraction*, 2018. arXiv: 1804.09364 [cs.RO].

[176] S. Shalev-Shwartz, S. Shammah, and A. Shashua, *Safe, multi-agent, reinforcement learning for autonomous driving*, 2016. arXiv: 1610.03295 [cs.AI].

[177] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577. DOI: 10.1109/CVPR.2018.00376.

[178] D. Lee, Y. P. Kwon, S. McMains, and J. K. Hedrick, "Convolution neural network-based lane change intention prediction of surrounding vehicles for acc," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317874.

[179] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016. DOI: 10.1109/TIV.2016.2578706.

[180] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond - status and future of the de facto standard for the description of road networks," in *Proceedings of the Driving Simulation Conference Europe*, 2010.

[181] World Health Organization, *Global Status Report on Road Safety 2018*. 2019.

[182] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022. DOI: 10.1109/TRO.2022.3173711.

[183] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, 2023. DOI: 10.1126/scirobotics.adg1462.

[184] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[185] P. Newman and P. Corke, "Editorial: Data papers—peer reviewed publication of high quality data sets," *The international journal of robotics research*, vol. 28, no. 5, pp. 587–587, 2009. DOI: 10.1177/0278364909104283.

[186] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013. DOI: 10.1177/0278364913491297.

[187] K. Sun, K. Mohta, B. Pfrommer, *et al.*, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018. DOI: 10.1109/LRA.2018.2793349.

[188] W. Zhao, A. Goudar, X. Qiao, and A. P. Schoellig, *Util: An ultra-wideband time-difference-of-arrival indoor localization dataset*, 2024. arXiv: 2203.14471 [cs.RO].

[189] *AlphaPilot – Lockheed Martin AI Drone Racing Innovation Challenge*, https://www.herox.com/alphapilot/teams, 2019.

[190] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021. DOI: 10.1126/scirobotics.abg5810.

[191] P. Foehn, D. Brescianini, E. Kaufmann, *et al.*, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022. DOI: 10.1007/s10514-021-10011-y.

[192] M. Burri, J. Nikolic, P. Gohl, *et al.*, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016. DOI: 10.1177/0278364915620033.

[193] A. L. Majdik, C. Till, and D. Scaramuzza, "The Zurich urban micro aerial vehicle dataset," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 269–273, 2017. DOI: 10.1177/0278364917702237.

[194] C. Pfeiffer and D. Scaramuzza, "Human-piloted drone racing: Visual processing and control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3467–3474, 2021. DOI: 10.1109/LRA.2021.3064282.

[195] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "NeuroBEM: Hybrid aerodynamic quadrotor model," in *Robotics: Science and Systems XVII*, Robotics: Science and Systems Foundation, 2021. DOI: 10.15607/rss.2021.xvii.042.

[196] P. Foehn, E. Kaufmann, A. Romero, *et al.*, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, eabl6259, 2022. DOI: 10.1126/scirobotics.abl6259.

[197] C. de Wagter, F. Paredes-Vallés, N. Sheth, and G. C. de Croon, *The artificial intelligence behind the winning entry to the 2019 ai robotic racing competition*, 2021. arXiv: 2109.14985 [cs.RO].

[198] E. Kaufmann, M. Gehrig, P. Foehn, *et al.*, "Beauty and the beast: Optimal methods meet learning for drone racing," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 690–696, 2018. DOI: 10.1109/ICRA.2019.8793631.

[199] MultiWii, *Multiwii serial protocol*, http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol.

[200] Betaflight, *The betaflight open source flight controller firmware project*, https://github.com/betaflight/betaflight.

[201] MultiGP, *Multigp drone racing gate*, https://www.multigp.com/product/drone-racing-gate-bundle/.

[202] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2018. DOI: 10.1109/LRA.2017.2776353.

[203] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8. DOI: 10.1109/IROS.2018.8593739.

[204] H. X. Pham, H. I. Ugurlu, J. Le Fevre, D. Bardakci, and E. Kayacan, "Deep learning for vision-based navigation in autonomous drone racing," *Deep Learning for Robot Perception and Cognition*, pp. 371–406, 2022. DOI: 10.1016/B978-0-32-385787-1.00020-8.

[205] K. Chen, J. Wang, J. Pang, *et al.*, *Mmdetection: Open mmlab detection toolbox and benchmark*, 2019. arXiv: 1906.07155 [cs.CV].

[206] M. Contributors, *Openmmlab pose estimation toolbox and benchmark*, https://github.com/open-mmlab/mmpose, 2020.

[207] *COCO - Common Objects in Context - Data format*, https://cocodataset.org/#format-data.

[208] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '85, ACM, 1985, pp. 245–254. DOI: 10.1145/325334.325242.