Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Energy Engineering (38^{th}cycle)

Learning-Based Control of Mobility and Automotive Systems Across Scales: From Fleet Coordination to Engine Combustion Modeling

By

Luigi Tresca

******

Supervisor(s):
Prof. Federico Millo
Dr. Andrea Piano
Prof. Luciano Rolando

Doctoral Examination Committee:
Prof. A.B. , Referee, University of...
Prof. C.D, Referee, University of...

Politecnico di Torino
2026

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Luigi Tresca
2026

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents

Abstract

Road transport is a major source of greenhouse-gas emissions, and the European Union legislation framework raises fleet CO_2 targets on the way to a 2035 zero-emission sales target, pushing both for vehicle efficiency and to find fleet-level solutions. This dissertation is focused on developing efficient controls and models for the automotive and mobility sectors, employing machine learning at three linked scales: control of urban fleets and hybrid vehicle energy management, and modeling the combustion process of internal combustion engines. At the fleet scale, Autonomous Mobility on Demand is posed as a network-flow control problem and addressed with a three-step framework, optimal dispatch, minimum-cost rebalancing, and a graph-aware Soft Actor–Critic policy that predicts region-level vehicle allocations. In an in-the-loop mesoscopic SUMO model of Luxembourg, this maximizes the operational profit for the operator, while reducing waiting time for passengers and fleet-level emissions, and remains robust across time of day, spatial aggregation, and simulator fidelity. At the vehicle scale, learning-based energy management systems are developed for a state-of-the-art plug-in hybrid electric vehicle and benchmarked against optimization strategies, such as dynamic programming and equivalent consumption minimization strategy. A dynamic programming-trained LSTM and an off-policy Soft Actor-Critic are trained on a backward model and validated on a detailed vehicle virtual test rig in both charge sustaining and charge depleting operation; they deliver competitive fuel/CO_2 results, respect SoC constraints, and generalize across cycles and initial conditions. At the combustion scale, two real-time data-driven models are introduced: a neural single-Wiebe parameterization for burn rate prediction, and a hybrid recurrent model that reconstructs the full burn rate profile and stays stable across speed, load, and dilution sweeps. Together, these results show that embedding domain knowledge in learning methods enables scalable, real-time decisions, from fleet rebalancing to power split and combustion prediction, supporting near-term emission cuts while meeting tightening policy targets.

Contents

List of Figures

List of Tables

List of Symbols

Acronyms

AC	Actor-Critic
ACC	Adaptive Cruise Control
AI	Artificial Intelligence
AMOD	Autonomous Mobility on Demand
BEV	Battery Electric Vehicle
BSFC	Brake Specific Fuel Consumption
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CTM	Cell Transmission Model
DDCA	Dual Dilution Combustion Approach
DDPG	Deep Deterministic Policy Gradient
DPG	Deterministic Policy Gradient
DDQL	Double Deep Q-Learning
DP	Dynamic Programming
ECM	Equivalent Circuit Model
ECMS	Equivalent Consumption Minimization Strategy
EGR	Exhaust Gas Recirculation
EM	Electric Motor
EMS	Energy Management System
EV	Electric Vehicle
GDI	Gasoline Direct Injection
GHG	Greenhouse Gas
GNN	Graph Neural Network
GPR	Gaussian Process Regression

GPU	Graphics Processing Unit
HEV	Hybrid Electric Vehicle
HVAC	High Voltage Air Compressor
ICE	Internal Combustion Engine
IDM	Intelligent Driver Model
ITE	Indicated Thermal Efficiency
LP	Linear Programming
LSTM	Long Short-Term Memory
LWR	Lighthill–Whitham–Richards
MFB10	10% Mass Fraction Burned
MFB50	50% Mass Fraction Burned
MFB10-75	10% to 75% Mass Fraction Burned
MIP	Mixed-Integer Programming
MIQP	Mixed-Integer Quadratic Programming
ML	Machine Learning
MPC	Model Predictive Control
NEDC	New European Driving Cycle
NN	Neural Network
OCV	Open Circuit Voltage
pHEV	plug-in Hybrid Electric Vehicle
PID	Proportional–Integral–Derivative
PMP	Pontryagin's Minimum Principle
QL	Q-Learning
QP	Quadratic Programming
RB	Rule-Based
RDE	Real Driving Emissions
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SAC	Soft Actor–Critic
SDP	Stochastic Dynamic Programming
SI	Spark Ignition
SoC	State of Charge
TD3	Twin Delayed Deep Deterministic Policy Gradient
VNT	Variable Nozzle Turbine
VVA	Variable Valve Actuation

WLTC	Worldwide harmonized Light vehicles Test Cycle
WLTP	Worldwide harmonized Light vehicles Test Procedure

Chapter 1

Introduction

Over recent decades, the risks associated with climate change and resource depletion have become evident, prompting a broad policy response to cut and control Greenhouse Gas (GHG) emissions. In Europe, the European Green Deal sets a course toward a modern and resource-efficient economy with net-zero emissions by 2050, complemented by the Fit for 55 package that targets at least a 55% reduction by 2030 relative to 1990 levels [6]. The transportation sector is deeply involved in this transition and accounts for roughly 35% of global energy use [7]. Within the Fit for 55 package, the EU targets to tighten fleet-average CO_2 emissions standards for light-duty vehicles: the 2030 emissions cut target for new passenger cars rises from 37.5% to 55% and for new vans from 31% to 50%, both relative to the 1990 reference level; moreover, from 2035 onward the standard requires a 100% reduction for newly registered cars and vans. At the same time, the increase in urbanization concentrates the location of vehicles and so the source of pollution. More than half of the world's population already lives in cities, and this share is projected to reach 60% by 2050 [8]. In dense urban areas, congestion, energy consumption, and pollution intensify, with road traffic linked to about thirty percent of urban emissions and fifty percent of health-related costs due to car-generated air pollution [9]. Effective decarbonization demands coordinated action at two interconnected scales. At the vehicle level, powertrains must achieve higher efficiency and lower carbon footprint. At the system level, mobility must be delivered through smarter, energy-aware operations, most notably via intelligently managed shared fleets. Such fleets can minimize empty mileage and idle time through optimized dispatching, routing, and rebalancing, while individual vehicles adopt energy-efficient driving

and charging strategies. Together, these measures ensure that each vehicle-kilometer is served with a smaller environmental footprint.

At the fleet scale, Autonomous Mobility on Demand (AMoD) is a promising urban mobility architecture: a centrally coordinated fleet of connected, self-driving vehicles that serves on-demand trips. By optimally assigning requests and proactively rebalancing idle vehicles, AMoD can increase vehicle utilization, reduce parking demand, and still preserve point-to-point convenience [10, 11]. Realizing these benefits, however, depends on the quality of coordination. Suboptimal repositioning or matching can inflate passenger waiting times and inject empty vehicles into already saturated corridors [12]. The underlying control problem is inherently spatiotemporal, stochastic, and high-dimensional, with strong network couplings and feedback from congestion and time-varying demand. These characteristics pose substantial challenges, balancing operator performance, limiting environmental impacts, and meeting strict real-time computational requirements, within the non-stationary traffic environments in which such fleets must operate.

At the vehicle level, one of the most promising solutions to lower emissions is represented by vehicle powertrain electrification. Policy incentives and expanding product ranges have accelerated the uptake of Electrified Vehicles (EVs) in Europe and worldwide [13–15]. Battery Electric Vehicles (BEVs) are increasingly popular, yet further improvements in performance, cost, and durability are still needed to reach parity with conventional vehicles. Limited range, long charging times, and a still not developed charging infrastructure sustain the mid-term role of Hybrid Electric Vehicles (HEVs) and plug-in Hybrid Electric Vehicles (pHEVs), which combine desirable features of electric and conventional powertrains [16]. The efficiency of hybridization depends on how energy is managed. The Energy Management System (EMS) must schedule engine and motor torque split and battery power so that power usage is optimized under operational and physical constraints [17]. Computer-aided simulation tools are widely used to evaluate strategies, and a substantial body of research has pushed the field toward methods that are implementable in real time yet remain close to optimal [18, 19].

A third, complementary scale concerns predictive combustion modeling. Even in an electrifying fleet, many electrified powertrains will still rely on an Internal Combustion Engine (ICE) for years to come, and extracting the highest efficiency from that engine is essential to reduce fuel use and emissions. Achieving this requires

combustion models that are accurate across operating conditions and fast enough for real-time use in virtual test rigs, calibration loops, and embedded controllers [20, 21] in order to not wrongly estimate their consumption and emissions performance. High-fidelity multi-zone and turbulence–chemistry models can capture complex in-cylinder physics, but are too computationally demanding [22, 23] for these roles. Simple phenomenological laws, by contrast, are fast but may struggle to represent non-standard burn behaviors and transient effects with sufficient fidelity. Data-driven surrogates offer a middle path: learned models can approximate the combustion process with high accuracy while meeting strict timing budgets [24]. Embedding such surrogates within lightweight thermodynamic backbones enables combustion prediction, supports EMS development with more faithful engine responses, and helps translate electrification into consistent real-world savings whenever the engine is active.

Within this policy and technological landscape, this dissertation advances a structure-plus-learning agenda across three interconnected scales so that efficiency gains compound from fleets to vehicles and down to combustion. At the city scale, AMoD coordination is formulated as a network-flow control task in which a graph-encoded Reinforcement Learning (RL) policy proposes region-level rebalancing actions and a minimum-cost flow layer enforces optimality in matching and rebalancing distribution-to-flow translation. In a calibrated mesoscopic study, this hierarchy approaches Model Predictive Control (MPC) performance while reducing passenger waiting time and containing overall fleet emissions, showing that benefits arise from how and when vehicles are repositioned and favoring predictive over reactive strategies. At the vehicle scale, learning-based EMSs are developed and evaluated against Dynamic Programming (DP) upper bounds and Equivalent Consumption Minimization Strategy (ECMS) baselines on a detailed virtual test rig of a pHEV. At the combustion scale, two data-driven surrogates are constructed. A Neural Network (NN) model the parameters of a single-Wiebe law offers a fast, compact description of the main burn, while a hybrid recurrent model predicts a robust full burn rate able to correctly predict the full combustion process while remaining stable across sweeps in speed, load, and dilution, achieving superior performance in terms of robustness and generalization.

Together, these contributions illustrate how domain structure and learning can be integrated to deliver real-time, scalable decisions for coordinated fleets, energy-

aware vehicles, and efficient engines, aligning operational performance with the environmental objectives that motivate today's regulatory trajectory.

The remainder of the dissertation is structured as follows:

- Chapter 1 locate the research within climate policy and the European regulatory pathway, surveys electrification trends and their implications for industry and cities, and motivates the need for coordinated fleets and effective control, outlining the thesis scope and layout.
- Chapter 2 reviews the Artificial Intelligence background with emphasis on NNs and RL, clarifies problem formulation and algorithm families, and focuses on Double Deep Q-Learning (DDQL) and Soft Actor–Critic (SAC) as representative methods for long-horizon decision making.
- Chapter 3 develops control applications across scales. It first establishes modeling foundations for traffic simulation at macro, meso, and micro fidelity, introduces xEV and powertrain models, and summarizes optimal control, optimization-based, heuristic, and learning-based methods. It then presents Scale One on AMoD coordination and Scale Two on energy management of hybrids, covering the case study and methodology, charge sustaining and charge depleting policies, performance on a detailed virtual test rig, and evidence of generalization.
- Chapter 4 investigates modeling applications to predictive combustion. It introduces phenomenological and data-driven combustion models, details the case study and methodology, including feature selection and distribution analysis, NN-Wiebe calibration, and a hybrid recurrent model for burn-rate prediction, and reports results for both the MLP-Wiebe and Hybrid-RNN approaches, followed by conclusions and future directions.
- Chapter 5 summarizes the main findings, discusses limitations, and outlines priorities for future work.

Chapter 2

Artificial Intelligence Background

In recent decades, remarkable advancements in computational technologies, particularly the exponential growth in Central Processing Unit (CPU) and Graphics Processing Unit (GPU) performance and the flexible scalability offered by cloud computing, have significantly accelerated the integration and application of Artificial Intelligence (AI) across diverse scientific and engineering disciplines. Within this broad spectrum, AI-driven methodologies designed for optimizing complex physical and networked systems have emerged prominently, demonstrating measurable improvements in both operational performance and system reliability. The transportation sector, in particular, has benefited substantially from such AI advancements, with successful implementations ranging from predictive modeling of transportation demand [25], through sophisticated fleet management strategies in networked mobility systems [1, 26], down to more granular applications such as powertrain control strategies within HEVs [2], and even detailed modeling of physical phenomena like combustion processes in ICE [27, 28]. Although AI is an expansive field encompassing numerous approaches and methodologies, a particularly insightful definition was articulated by Nilsson (2009):

"Artificial Intelligence is that activity devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment" [29].

At the core of AI lies Machine Learning (ML), a foundational discipline initially propelled by Samuel's groundbreaking research (1959). Samuel demonstrated how computational systems could independently develop strategic competencies through

iterative learning, exemplified by programs capable of mastering games like checkers. ML techniques are fundamentally aligned with established scientific methodologies, heavily leveraging inductive reasoning to generate hypotheses through pattern recognition in empirical data. Unlike deductive reasoning, where conclusions are logically drawn from established premises, inductive reasoning creates generalizations based on observed instances. Consequently, inductive inferences are inherently provisional and remain open to revision upon encountering new data. A classic illustration of this principle involves observing swans: an ML model analyzing a dataset consisting entirely of white swans may hypothesize that all swans are white. However, such an inference is tentative, subject to immediate revision upon the discovery of even a single black swan. Despite this inherent uncertainty, inductive reasoning based on extensive, high-quality datasets holds substantial practical significance, particularly given today's vast data availability and computational power.

Classification of Machine Learning Algorithms

ML encompasses a diverse spectrum of algorithmic approaches, and selecting an optimal technique inherently depends on various problem-specific characteristics, such as dataset structure, the nature of the variables, and the predictive goals. While numerous classification schemes for ML exist in the literature [30], a widely recognized framework categorizes these methodologies into three principal groups [31]:

- **Supervised Learning:** Algorithms in this category derive predictive relationships between inputs (independent variables) and outputs (dependent variables) using labeled datasets. Typically, data is partitioned into training and testing subsets, allowing algorithms to identify and internalize patterns from training samples and subsequently evaluate their predictive capabilities on the test set. Common supervised learning methods include Regression Analysis, Decision Trees, Random Forests, K-Nearest Neighbors (K-NN), Logistic Regression, and Gaussian Process Regression (GPR).
- **Unsupervised Learning:** These algorithms aim to uncover intrinsic structures and hidden patterns within unlabeled datasets. In contrast to supervised learning, unsupervised approaches do not rely on predefined labels or correct

outcomes, but rather infer groupings, similarities, or latent patterns. Notable examples include K-Means Clustering and the Apriori algorithm.

- **Reinforcement Learning:** In RL, algorithms learn by interacting with their environment through trial and error, guided by a numeric reward system. Rather than receiving explicit instructions about the correctness of specific actions, RL systems autonomously discover optimal strategies to maximize cumulative rewards over time. This approach finds extensive applications in decision-making tasks where long-term optimization is crucial.

Building upon the above classification, this thesis places particular emphasis on supervised learning techniques designed for regression tasks, situations in which the output variable is continuous. A representative example is the prediction of instantaneous ICE torque required by power-split controllers in HEVs. Accurate torque estimation is critical for optimally distributing power between the electric motor and the ICE, thereby improving overall energy efficiency. Within the spectrum of regression algorithms, NNs have risen to prominence owing to their universal function approximation capability. The universal approximation theorem asserts that, given sufficient network depth and width, NNs can approximate any measurable function to an arbitrary degree of accuracy. This theoretical property, coupled with their empirical success in modeling highly nonlinear and multivariate relationships, renders NNs exceptionally well suited for both control applications and the detailed representation of complex physical processes examined in this research.

Chapter Structure

In the subsequent sections, this chapter will delve into how AI, specifically ML methodologies, can be effectively applied within various scales of transportation systems, providing the necessary background to understand the methodologies presented. After the classification of ML algorithms presented above, the discussion focuses on supervised regression methods, laying the ground on NNs in Section 2.1. From generic feed-forward networks (Section 2.1.1), the discussion flows through sequence-aware recurrent architectures (Section 2.1.2) to graph-centric models (Section 2.1.3), illustrating how growing structural complexity in data motivates increasingly sophisticated network designs. The chapter closes with an overview of

modern RL, linking its value- and policy-based formulations to the control challenges explored in subsequent case studies (Section 2.2).

2.1 Neural Networks

NNs have proven capable of solving a wide variety of problems, including pattern recognition, classification, clustering, and more [32]. They are inspired by the human brain, consisting of interconnected "neurons" that mimic the way biological neurons process information. Over the decades, researchers have developed different NN architectures to handle different types of data. In general, a feed-forward NN (also known as a fully connected network or multilayer perceptron) passes data through layers of neurons in one direction (input to output) and is the foundation of many deep learning models [33]. Convolutional Neural Networks (CNNs) [34] introduce convolution operations and are especially effective for grid-like data such as images, making them powerful for image and pattern recognition tasks. Recurrent Neural Networks (RNNs) [35], on the other hand, include feedback loops, allowing them to maintain an internal state and model sequential or time-series data (e.g. for forecasting future values in a sequence). More recently, Graph Neural Networks (GNNs) [36] have emerged to extend deep learning to graph-structured data (non-Euclidean data domains), where relationships are represented as nodes and edges rather than fixed grids or sequences. Together, these architectures enable NNs to tackle diverse data modalities and complex nonlinear relationships. NNs' flexibility and learning capability have led to their adoption in many fields. For instance, in transportation systems, deep networks (including RNNs and GNNs) are used to forecast traffic conditions and optimize network flows [37]. In advanced control strategies (such as vehicle engine control or robotics) and combustion modeling, NNs serve as powerful function approximators for system dynamics and chemical processes. Several application in the context of EMS for HEVs have been emerged in the last years, both applying NNs to directly mimic the policy of optimal control strategies [38, 39] using RNNs, or by implementing Deep Reinforcement Learning (DRL) policies to directly approximate the optimal policy from interaction with the vehicle [2, 40], a class of algorithms described in Section 2.2. Additionally, by training on data, they can capture patterns for tasks like engine combustion monitoring, on-board diagnostics, and predictive control, which are effectively modeled with physics-based models,

but at the cost of high computational requirements. With data-driven approach it is possible to reduce the computational demand without greatly affecting the performance if the model is properly trained [27]. These examples underscore the broad applicability of NNs in solving complex problems across transportation, control, and combustion domains.

2.1.1 Feed-Forward Neural Networks

The fundamental building block of a NN is the artificial neuron (depicted in Figure 2.1), often implemented as a perceptron. The perceptron, introduced by Rosenblatt in 1958 [41], computes a weighted sum of its inputs and then applies an activation function to produce an output. Mathematically, a perceptron with inputs $x_1, x_2, \ldots, x_N$ produces an output y as follows:

$$y = \sigma(W \cdot \mathbf{x} + b) \tag{2.1}$$

where $W = [w_1, w_2, \ldots, w_N] \in \mathbb{R}^N$ are the weights on each input, $\mathbf{x} = [x_1, \ldots, x_N]^T \in \mathbb{R}^N$ is the input vector, b is a bias term, and $\sigma(\cdot)$ is the activation function, which allows to consider nonlinearities in the function approximation process.

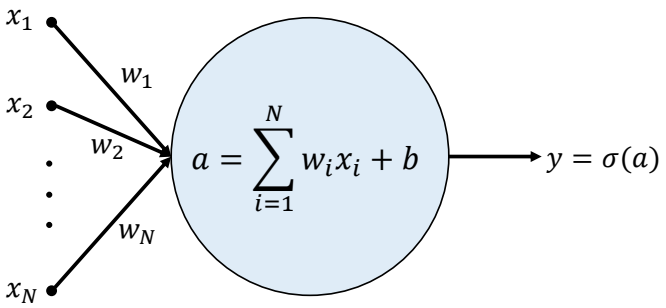


Fig. 2.1 Perceptron layout.

In a feed-forward NN, many neurons are organized into layers: an input layer, one or more hidden layers, and an output layer. Each neuron in a layer takes as input the outputs of the previous layer's neurons (or the raw inputs for the first layer). The

neurons are fully connected between adjacent layers, hence the term fully connected network. The output of each neuron in one layer becomes an input to neurons in the next layer, and information flows forward only, making it a feed-forward architecture. After the weighted sum at each neuron, an activation function is applied to introduce nonlinearity (common choices include the sigmoid logistic function, hyperbolic tangent, or the Rectified Linear Unit). By stacking multiple layers, feed-forward networks, also called Multi Layer Perceptrons (MLPs), can model more complex nonlinear decision boundaries.

During the training of an NN, the goal is to automatically adjust the weights W and biases b so that the network's output y matches the desired output for each input, by minimizing the prediction error on the training set (usually expressed as a Mean Squared Error (MSE) between predicted and target output). This learning is typically achieved by the *back-propagation* algorithm [42] combined with an optimization method like gradient descent [43], or its more sophisticated version, as Adam [44]. Back-propagation efficiently computes the gradient of the loss function with respect to each weight, enabling the network to learn by incrementally updating weights in the direction that reduces error. In essence, the network "learns" a highly nonlinear input–output relationship by iteratively tuning its parameters to minimize the difference between its predictions and the ground truth, which is represented by the target outputs in the training dataset. The concept of back-propagation was first formulated in the 1970s [45], but it become popular in the mid-1980s when Rumelhart, Hinton, and Williams popularized its use for training multilayer networks, demonstrating that it could effectively train deep neural architectures [42]. As a result Deep Neural Networks (DNNs), essentially feed-forward NNs with many hidden layers, became feasible and started to outperform shallow network models on complex tasks. The transition from shallow to deep architectures enabled the mapping of much more complex and highly nonlinear functions that shallow networks could not efficiently represent [33]. Several factors catalyzed the rise of deep feed-forward networks in recent decades. The advent of cheaper, high-performance parallel computing, especially the GPU, significantly accelerated the training of large networks, which involve millions of weight updates. In addition, the availability of massive datasets ("big data") provided the raw material needed for deep networks to generalize well. These developments, coupled with algorithmic improvements and regularization techniques, have led to the modern deep learning revolution. Today's deep feed-forward networks form the basis of many applications,

from computer vision and natural language processing to system modeling and control, wherever a complex functional mapping from inputs to outputs is required.

2.1.2 Long-Short Memory Layer

While feed-forward NNs do not use feedback loop between inputs, many real-world problems involve sequential data where past inputs influence future ones, like in time series, sentences, control signals over time, and many others. RNNs [35] addresses this limitation by introducing loops in the network, as depicted in Figure 2.2: the output of a neuron at one time step is fed back as input in the next time step. This gives RNNs an inherent memory and process sequences of inputs of arbitrary length.

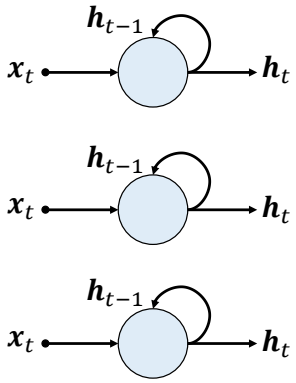


Fig. 2.2 RNN layer layout.

However, early RNNs suffered from the vanishing (and exploding) gradient problem [46], as one tries to backpropagate error gradients through many time steps, the gradients can decay or blow up, making it difficult for standard RNNs to learn long-range dependencies in sequences.

Long Short-Term Memory (LSTM) networks were proposed by Hochreiter and Schmidhuber (1997) to overcome these issues [47]. An LSTM is a type of RNN that embeds a special unit called a memory cell along with gating mechanisms that regulate the flow of information. Each LSTM cell contains three gates, often called the forget gate, input gate, and output gate, which control what information is kept in memory, what new information is added, and what information is output at each

time step, as depicted in Figure 2.3. The gating is implemented through sigmoid activations (which output values between 0 and 1, functioning like "valves" for information) and element-wise multiplication to selectively pass information. The LSTM's equations for a single time step t can be written as follows:

$$\begin{cases} i_t = \sigma_g(W_i \cdot \mathbf{x}_t + U_i \cdot \mathbf{h}_{t-1} + b_i) \\ f_t = \sigma_g(W_f \cdot \mathbf{x}_t + U_f \cdot \mathbf{h}_{t-1} + b_f) \\ o_t = \sigma_g(W_o \cdot \mathbf{x}_t + U_o \cdot \mathbf{h}_{t-1} + b_o) \\ \tilde{c}_t = \sigma_c(W_c \cdot \mathbf{x}_t + U_c \cdot \mathbf{h}_{t-1} + b_c) \\ \mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{c}_t \\ \mathbf{h}_t = o_t \odot \sigma_c(\mathbf{c}_t) \end{cases} \tag{2.2}$$

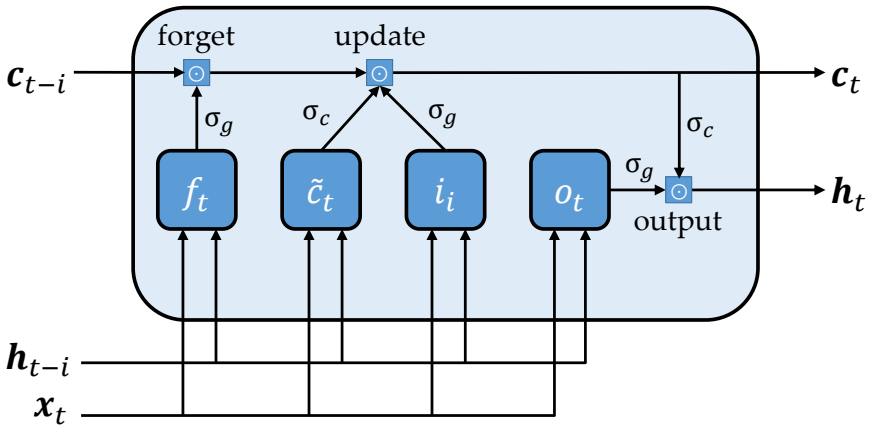


Fig. 2.3 LSTM layer layout: the predicted output patterns are assigned to the training set (the observations).

where $\mathbf{x}_t$ is the input at time t, $\mathbf{h_{t-1}}$ is the previous hidden state, and $\mathbf{c}_{t-1}$ is the previous cell state. $\sigma_g(\cdot)$ denotes the sigmoid function (for gates), $\sigma_c(\cdot)$ is usually a tanh activation that generates candidate values, and $\odot$ is element-wise multiplication. In these equations, i_t, f_t, o_t are the input, forget, and output gate activations respectively, each between 0 and 1 due to the sigmoid activation, and $\tilde{c}_t$ is the candidate update to the cell state. The cell state c_t is updated by linearly combining the previous cell state c_{t-1}, scaled by f_t, which decides how much to forget or retain from the past, with the new candidate content $\tilde{c}_t$, scaled by i_t,

which decides how much new information to transfer in the new cell state. The new hidden state h_t, which is also the output of the LSTM at time t, is then given by the transformed cell state (passed through tanh) filtered by the output gate o_t. Essentially, the forget gate f_t controls the cell's memory clearance, the input gate i_t controls how much new information flows into the memory, and the output gate o_t controls how much of the memory is revealed to the next layer or as output. This gated design allows LSTMs to maintain and update a memory over long sequences, thereby remembering long-term dependencies far better than standard RNNs. By preventing older important information from being completely overwritten (thanks to forget gates) and by selectively adding new information, LSTMs mitigate the vanishing gradient problem. In practice, LSTM networks have been extraordinarily successful in sequence modeling tasks. They have been used extensively in language modeling, speech recognition, and time-series forecasting. For example, LSTM-based models can learn temporal patterns in traffic flow data to improve short-term traffic predictions [48], and they are used in control systems to predict time-dependent signals (such as the control variables in optimal control where the evolution of input signals matters). The ability of LSTMs to handle time dependencies makes them well-suited for any application where the context through time matters significantly.

The application of is applied in both the control and modelling application. In fact, a power-split controller for a pHEV was implemented by training an LSTM network with a dataset built with the optimal control trajectory of DP, as reported in [38] (see Section 3.4.2). A hybrid network, combining an LSTM layer with a series fully-connected layer was then implemented to model the burn rate of a highly diluted ICE in the modelling implementation section, as reported in Section 4.3. LSTM networks are exploited for both control and system modelling. First, a power-split energy management controller for a pHEV was derived by training an LSTM network on a data set generated from the DP optimal control trajectory, as detailed in [38] (Section 3.4.2). Subsequently, a hybrid architecture comprising an initial LSTM layer that works in parallel with a series of fully connected layers was designed to predict the burn rate of a strongly diluted ICE within the modelling application framework (Section 4.3).

2.1.3 Graph Neural Networks

The aforementioned NNs (fully-connected, convolutional, recurrent) generally operate on regular structured data like vectors, grids, or sequences. However, many problems involve data that are more naturally expressed as graphs, sets of entities with complex relationships. Examples include transportation networks, social networks, communication networks, molecular structures, knowledge graphs, and many others. GNNs are a class of deep learning models designed to directly work with graph-structured data [36]. In a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, data is organized as nodes $\mathscr{V} = \{v_i\}_{i=1:N_v}$ connected by edges $\mathscr{E} = \{e_k\}_{k=1:N_e} \subseteq \mathscr{V} \times \mathscr{V}$, and each node, and possibly each edge, can have features. Connectivity in the graph is encoded by the adjacency matrix $A \in \mathbb{R}^{N_v \times N_v}$:

$$A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathscr{E}, \\ 0, & \text{otherwise}, \end{cases} \tag{2.3}$$

while the diagonal degree matrix $D \in \mathbb{R}^{N_v \times N_v}$ contains information about the degree of each node, that is, the number of edges attached to each node.

$$D_{ij} = \begin{cases} \sum_{j=1}^{N} A_{ij}, & i = j, \\ 0, & i \neq j, \end{cases} \tag{2.4}$$

Both A and D are typically augmented with self-loops via $\hat{A} = A + I$ and $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. The key idea of GNNs is to iteratively update each node's representation by aggregating information from its neighbors in the graph. This process, often called message passing [49] or graph convolution, allows the network to learn embeddings that capture both the features of each node and the structure of its neighborhood. One popular formulation is the Graph Convolutional Network (GCN) proposed by Kipf and Welling (2016) [50], which performs an aggregation analogous to a convolution on the graph. In a GCN layer, the feature vector of each node $\mathbf{x}_i \in R_f^N$ (with N_f the number of node features) is updated by taking a weighted sum of the feature vectors of its neighbors (and itself), followed by a non-linear transformation. A representation of the aggregation mechanism of GCN is depicted in Figure 2.4.

Mathematically, a GCN update can be written in matrix form as:

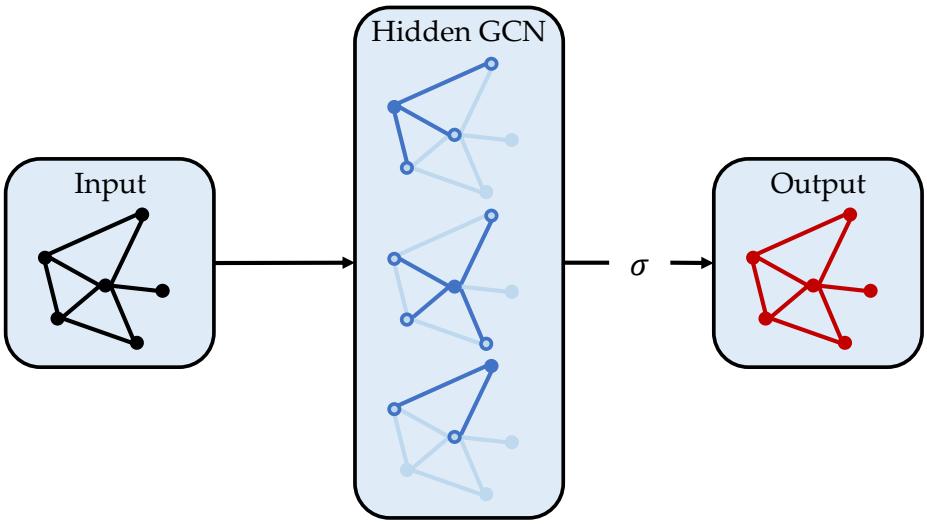


Fig. 2.4 LSTM layer layout: the predicted output patterns are assigned to the training set (the observations).

$$X' = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW\right) \tag{2.5}$$

where $X \in R^{N_v \times N_f}$ is the matrix of node features (each row corresponds to a node's feature vector), W is a trainable weight matrix, and $\sigma(\cdot)$ is an activation function (like ReLU) applied element-wise. Here $\hat{A} = A + I$ is the adjacency matrix of the graph with added self-connections (so that each node includes its own features in the aggregation), and $\hat{D}$ is the diagonal degree matrix of $\hat{A}$. The term $\hat{D}^{-1/2}, \hat{A}, \hat{D}^{-1/2}$ is a normalized adjacency matrix, which scales the neighbor contributions by the inverse square root of their degrees, ensuring that the feature aggregation is properly normalized. In effect, this equation says: each node's new feature $\mathbf{x}_i'$ is obtained by averaging its own and its neighbors' current features $\mathbf{x}_j$ (with normalization) and then applying a linear transformation W and nonlinearity σ. Repeating this for multiple layers allows information to propagate over the graph: a node's representation can encode not just its immediate neighbors but neighbors further away, as successive layers aggregate more distant signals. By using such neighbor-aggregation schemes, GNNs can learn rich representations that incorporate graph topology and features. They have achieved impressive results in many domains where relational inductive bias is important. For example, in a transportation network (which naturally forms a graph of intersections and roads), GNN models can capture the spatial dependencies of traffic flows; this leads to more accurate traffic forecasting and better routing

or signal control strategies [51, 52]. Recent work shows GCN policies boost fleet rebalancing for AMoD systems [1, 53] and, when combined with graph-aware RNN models, also learn temporal demand patterns [54]. Building on these advances, in this work a GCN policy is embedded in an RL framework for AMoD coordination, training it in the SUMO traffic simulator [55] and validating robustness across times of day, demand patterns, and simulators' fidelity, thereby demonstrating that exploiting graph structure yields superior control of complex transportation networks.

2.2 Reinforcement Learning

RL [56] is a powerful, data-driven approach aimed at optimizing control policies for dynamic systems through direct interaction with their environments. Unlike classical DP, which typically demands exhaustive exploration of the entire state-action space and precise knowledge of future conditions (see Section 3.2.2), RL iteratively refines policies based on experiences gained from real-time interactions. This iterative approach significantly mitigates the *curse of dimensionality*, making RL particularly suitable for practical applications involving complex, high-dimensional real-world environments.

Formally, RL is framed within the mathematical structure of a Markov Decision Process (MDP), defined as follows:

$$M = (\mathscr{X}, \mathscr{U}, P, d_0, r, \gamma) \tag{2.6}$$

In this framework, $\mathscr{X}$ denotes the set of states $\mathbf{x} \in \mathscr{X}$, which can be discrete or continuous. Similarly, $\mathscr{U}$ represents the set of control actions $\mathbf{u} \in \mathscr{U}$. The dynamics governing state transitions are captured by the conditional probability $P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, with an initial state distribution $d_0(\mathbf{x}_0)$. The reward function $r : \mathscr{X} \times \mathscr{U} \to \mathbb{R}$ offers immediate feedback, informing the learning agent about the effectiveness of its actions in specific states. The agent's goal is to maximize the cumulative sum of rewards, adjusted by a discount factor $\gamma \in (0, 1]$ that balances immediate versus future rewards. Critically, the careful engineering of the reward function and appropriate selection of γ are essential yet challenging aspects of RL, greatly influencing the resulting policies. For instance, in EMS for HEVs, balancing fuel efficiency against

battery usage illustrates the sensitivity of the policy to reward shaping, generating a spectrum of strategies from conservative battery management to aggressive depletion.

The objective of RL algorithms is to find the optimal policy $\pi(\mathbf{u}_t|\mathbf{x}_t)$, defining the agent's decision-making strategy by mapping states to actions. During training, the agent collects *experiences*, represented as tuples $\{\mathbf{x}_t, \mathbf{u}_t, r_t, \mathbf{x}_{t+1}\}$, forming trajectories τ, which serve as the basis for iterative policy refinement. The probability distribution over these trajectories, given the policy $\pi(\mathbf{u}_t|\mathbf{x}_t)$, is expressed as:

$$p_\pi = d_0(\mathbf{x}_0)\prod_{t=0}^{H}\pi(\mathbf{u}_t|\mathbf{x}_t)P(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t) \tag{2.7}$$

Thus, the formal objective of RL becomes:

$$J_\pi = \mathbb{E}_{\tau\sim p_\pi}\left[\sum_{t=0}^{H}\gamma^t r(\mathbf{x}_t,\mathbf{u}_t)\right] \tag{2.8}$$

2.2.1 Classification of RL Algorithms

To effectively address diverse control challenges, various RL algorithms have been proposed, grouped according to their core features and underlying assumptions. These classifications facilitate selecting appropriate methods tailored to specific tasks and contexts.

Model-based vs Model-free Approaches

A primary distinction among RL algorithms depends on whether the environment's dynamics are explicitly modeled or not. *Model-based* approaches explicitly learn the transition dynamics $P(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)$ from interactions, subsequently using this learned model to simulate action outcomes and optimize policies offline. These approaches, such as Dyna [57], Model-Based Policy Optimization (MBPO) [58], and Dreamer [59], significantly enhance sample efficiency but can introduce biases due to modeling errors, potentially limiting their accuracy and robustness.

Conversely, *model-free* approaches avoid explicit modeling of the environment dynamics, directly deriving policies from trial-and-error interactions between the RL policy and the environment. Algorithms like SARSA, Q-Learning (QL) [60], and the

more advanced SAC [61] use this paradigm. Model-free methods typically excel in complex and high-dimensional environments due to their simplicity and avoidance of modeling errors, at the cost of a lower sample efficiency since there is no information regarding the response of the model to a control, rather than the reward feedback, requiring more policy updates during training compared to model-based approaches.

Discrete vs Continuous Environments

Another crucial categorization relates to the nature of state-action spaces. In *discrete* environments, state-action pairs constitute finite, countable sets, allowing methods like classical QL to employ tabular approaches effectively. However, many practical scenarios, especially physical or control systems, inherently possess *continuous* state-action spaces. Algorithms such as SAC and Deep Deterministic Policy Gradient (DDPG) [62] are specifically designed to handle continuous domains, often employing NNs to approximate policies and value functions, thus enabling more targeted and scalable control solutions.

Policy-based vs Value-based Approaches

RL methods also differ fundamentally in their optimization strategies.

Policy-based algorithms directly parameterize and optimize policies, typically using gradient-based methods. In these algorithms, a parametrized function, usually a NN, is used to model the policy that maps states to control actions, that can be deterministic $\pi(\mathbf{x}_t)$ or stochastic $\pi(\mathbf{u}_t|\mathbf{x}_t)$, also called the *actor*. Algorithms such as REINFORCE, Proximal Policy Optimization (PPO) [63], and Trust Region Policy Optimization (TRPO) [64] fall under this category, suitable for managing high-dimensional or continuous action spaces despite potential challenges in optimization stability. In fact, the policy gradient can be noisy and the algorithms are very sample inefficient, aspects that can cause the policy to get stuck in local minima.

Value-based focus on learning value functions and deriving a policy from them, rather than learning the policy directly. Two value functions can be learned: V^{π}, which maps the expected return associated with being in a particular state and then following the policy π:

$$V^{\pi}(\mathbf{x}) \;=\; \mathbb{E}_{\tau\sim p_{\pi}(\,\cdot\,|\mathbf{x}_0=\mathbf{x})}\left[\sum_{t=0}^{H}\gamma^{t}\, r(\mathbf{x}_t,\mathbf{u}_t)\right] \tag{2.9}$$

or the action-value function Q^{π}, which is defined as the expected cumulative return that the agent can collect taking an action $\mathbf{u}$ in the state $\mathbf{x}$ and then following a policy π.

$$Q^{\pi}(\mathbf{x},\mathbf{u}) \;=\; \mathbb{E}_{\tau\sim p_{\pi}(\,\cdot\,|\mathbf{x}_0=\mathbf{x},\,\mathbf{u}_0=\mathbf{u})}\left[\sum_{t=0}^{H}\gamma^{t}\, r(\mathbf{x}_t,\mathbf{u}_t)\right] \tag{2.10}$$

Key examples of model-free, value-based algorithms include the QL [60], one of the most common RL algorithms, and its derived versions, such as the Deep QL (DQL) [65] and the Double Q-Learning [66].

QL incrementally estimates the optimal action-value function by bootstrapping toward the reward plus the maximum predicted value of the next state, and works with a tabular approach in a discrete environment. Double QL uses the same update rule of QL, but introduces an independent value estimator so that one estimator selects the next-state action while the other evaluates it; this simple decoupling sharply reduces the over-estimation bias that arises from using the same values for both selection and evaluation, one of the main drawback of QL. Deep QL lifts QL to high-dimensional and continuous observation spaces, while still working with discrete action spaces, by replacing the tabular Q-function with an NN. With DQL, the "curse of dimensionality" problem that affects the QL is partially solved. Finally, Double DQL (2016) combines the bias-reduction idea of Double QL with the function-approximation power of DQL using two value estimator parametrized with NNs. The Double DQL is described in deep in Section 2.2.2, since it is exploited in the optimization of the energy management of a pHEV, whose results are presented in the lower-scale case study in Section 3.4.

Actor-Critic Methods

Actor-Critic (AC) algorithms bridge the strengths of policy- and value-based methods by simultaneously learning a parameterized policy $\pi_{\vartheta}(\mathbf{x}_t)$ (or $\pi_{\vartheta}(\mathbf{u}_t|\mathbf{x}_t)$ for a stochastic setting), called the *actor*, and an associated value function $Q_{\varphi}(\mathbf{x}_t,\mathbf{u}_t)$, also known as the *critic*. Training the actor with feedback from the critic mitigates optimization

issues, stabilizing policy updates and enhancing sample efficiency. Algorithms such as DDPG and SAC have emerged as state-of-the-art solutions for continuous control, balancing exploration and exploitation effectively through sophisticated value estimation and policy optimization techniques.

Deterministic Policy Gradient (DPG) algorithms [56] introduced the idea of learning a deterministic target policy while exploring with a separate stochastic behaviour policy. DDPG [62] extends DPG by representing both actor and critic with NNs. SAC [61] is the current state-of-the-art for continuous control, and maximizes a trade-off between expected reward and entropy (randomness) of the policy. By pursuing a maximum entropy objective, SAC encourages exploratory and diverse actions while learning. It employs two value estimators for more reliable value estimates and an adjustable temperature parameter to balance reward maximization against entropy. These innovations make SAC highly sample-efficient and robust, achieving excellent performance on continuous control benchmarks with stable convergence. Section 2.2.3 provides a detailed description of SAC and explains how it is applied in both the control scales: (i) to control the power-split of a pHEV using a stochastic policy, and (ii) to optimize the spatial distribution of idling vehicles.

On-policy vs Off-policy Methods

Finally, RL algorithms can be categorized based on experience utilization strategies. *On-policy* methods, including PPO and Advantage Actor–Critic (A2C, A3C), use experiences gathered directly from the current policy for updates, offering training stability but reduced sample efficiency. In contrast, *off-policy* methods, such as QL and its derived variants (Double QL and DQL), DDPG, and SAC, leverage past experiences collected by potentially different behaviour policies to update a different target policy, significantly enhancing sample efficiency. This flexibility, however, necessitates advanced stabilization techniques to mitigate the risk of unstable updates. QL-based algorithms collect experiences with an ε-greedy behaviour policy (i.e., choosing a random action with probability ε and the greedy action otherwise) and update the action-value function as if the greedy policy had been followed. Off-policy AC algorithms behave similarly: DDPG and SAC store transitions in a replay buffer and later sample mini-batches to update both actor and critic.

By clearly understanding these classifications, practitioners can strategically select and adapt RL methods suited to the specific requirements and constraints of

their target applications. Next sections detail a deeper presentation of the Double Deep QL (DDQL) (Section 2.2.2)) and the SAC (Section 2.2.3), which are employed as RL algorithms for the optimization of the control policies at both the scales considered.

2.2.2 Double Deep Q-Learning

Building upon the foundational distinctions established in the previous section regarding model-free, value-based RL algorithms, we now delve deeper into the DDQL method, a significant advancement over classical QL. As previously highlighted, QL stands out for its simplicity and effectiveness, working within discrete state-action spaces without explicitly modeling environment dynamics. QL employs Temporal Difference (TD) learning [67], which blends aspects of Monte Carlo (MC) [68] and DP [69]. TD learning acquires knowledge from direct environmental interactions similar to MC, and utilizes a bootstrapping technique similar to DP [70], estimating the Q-value function as defined earlier in Equation 2.10. Central to both DP and RL frameworks is the *Bellman equation*, which expresses a state's value (or state-action pair's Q-value) through immediate rewards and the next state's value. Through the Bellman equation, it is possible to find optimal decisions by breaking down complex decision-making problems into smaller, manageable steps, as follows:

$$Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = r(\mathbf{x}_t, \mathbf{u}_t) + \gamma \mathbb{E}_{\substack{\mathbf{x}_{t+1} \sim P \\ \mathbf{u}_{t+1} \sim \pi}} \left[Q^{\pi}(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) \right] \tag{2.11}$$

Here, $Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t)$ denotes the Q-value associated with policy π when selecting action $\mathbf{u}_t$ in state $\mathbf{x}_t$. The optimal policy's Q-value function, π^*, maximizes the expected future rewards, and its corresponding Bellman equation is formulated as:

$$Q^{\pi^*}(\mathbf{x}_t, \mathbf{u}_t) = \mathbb{E}_{\mathbf{x}_{t+1} \sim P} \left[r(\mathbf{x}_t, \mathbf{u}_t) + \gamma \max_{\mathbf{u}_{t+1} \in \mathscr{U}} Q(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) \right] \tag{2.12}$$

However, traditional value iteration approaches to solving Equation 2.12 suffer from scalability issues due to the complexity of extensive state-action spaces search. A more efficient approach leverages a parameterized Q-function $Q_{\varphi}(\mathbf{x}_t, \mathbf{u}_t)$, typically realized through an NN, directly estimating optimal Q-values from observed data. The TD update rule for this parameterized estimator is given by:

$$Q_\varphi(\mathbf{x}_t,\mathbf{u}_t) \leftarrow Q_\varphi(\mathbf{x}_t,\mathbf{u}_t) + \alpha \left[r(\mathbf{x}_t,\mathbf{u}_t) + \gamma \max_{\mathbf{u}_{t+1}\in\mathscr{U}} Q_\varphi(\mathbf{x}_{t+1},\mathbf{u}_{t+1}) - Q_\varphi(\mathbf{x}_t,\mathbf{u}_t) \right] \tag{2.13}$$

While effective, standard QL and DQL inherently face Q-value overestimation issue, arising from using the same parameters to select and evaluate actions, due to the application of the max operator. To address this limitation, DDQL [65] introduces an innovative decoupling strategy, employing two distinct NNs:

- **Behaviour network** Q_φ: Actively interacts with the environment, generating experiences and continually updating its parameters.
- **Target network** $Q_{\varphi'}$: Provides a stable evaluation reference, using periodically or soft updated parameters (see Equation 2.16).

By separating the action selection from its evaluation, DDQL significantly reduces the Q-value overestimation bias typically encountered in traditional QL. The TD target value $y(\mathbf{x}_t,\mathbf{u}_t)$ within DDQL becomes:

$$y(\mathbf{x}_t,\mathbf{u}_t) = r(\mathbf{x}_t,\mathbf{u}_t) + \gamma Q_{\varphi'}(\mathbf{x}_{t+1}, \operatorname*{argmax}_{\mathbf{u}_{t+1}\in\mathscr{U}} Q_\varphi(\mathbf{x}_{t+1},\mathbf{u}_{t+1})) \tag{2.14}$$

The training process and the separation between action selection and action evaluation are clearly depicted in Figure 2.5.

DDQL further enhances sample efficiency through an *experience replay buffer* $\mathscr{D}$, storing previous experiences $\{\mathbf{x}_t,\mathbf{u}_t,r_t,\mathbf{x}_{t+1}\}$, and then randomly sample mini-batches of $\mathscr{M}$ experiences from $\mathscr{D}$ for updates. The behaviour estimator Q_φ parameters are optimized by minimizing the loss function:

$$J_{Q_\varphi} = \mathbb{E}_{\{\mathbf{x}_t,\mathbf{u}_t,r_t,\mathbf{x}_{t+1}\}\sim\mathscr{D}} \left[\big(y(\mathbf{x}_t,\mathbf{u}_t) - Q_\varphi(\mathbf{x}_t,\mathbf{u}_t) \big)^2 \right] \tag{2.15}$$

Target network parameters are updated either through periodic synchronization to the behaviour parameters ($\varphi' = \varphi$) or via a soft-update rule to ensure stability:

$$\varphi' = \tau\varphi + (1-\tau)\varphi' \tag{2.16}$$

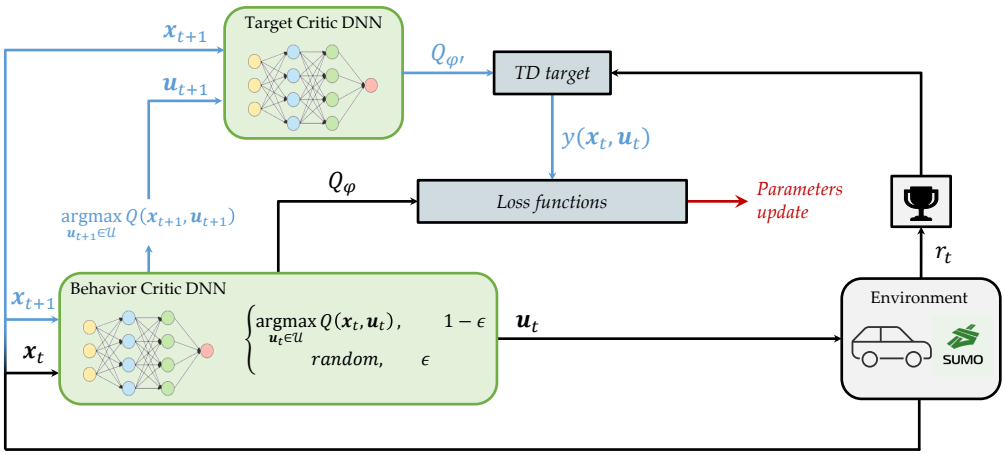


Fig. 2.5 Schematic representation of the architecture of the SAC agent.

This methodology substantially improves training stability, as the slowly evolving target network mitigates drastic fluctuations typical of rapid parameter updates. Thus, DDQL exemplifies a critical advancement within the broader landscape of off-policy, value-based RL methods, maintaining efficiency and stability while addressing significant limitations of its predecessors.

2.2.3 Soft Actor-Critic

While DDQL substantially advances traditional QL by addressing the overestimation bias and enhancing stability in discrete action spaces, it inherently struggles when confronted with complex, continuous state-action environments. This limitation motivates the introduction of AC algorithms, which unify the strengths of policy-based and value-based approaches discussed previously. Among these methods, the SAC [61] algorithm has emerged as a prominent and effective solution, particularly tailored for continuous control problems. SAC is characterized by three fundamental components: a distinct actor-critic structure involving separate policy (actor) and value function (critic) NNs, an off-policy training approach similar to DDQL to leverage previous experience efficiently, and entropy maximization to ensure robust exploration and policy stability. This entropy maximization distinguishes SAC from other RL algorithms, as it explicitly balances expected returns against policy randomness, leading to improved exploration efficiency and reduced risk of convergence to suboptimal solutions.

The SAC agent's training methodology, illustrated schematically in Figure 2.6, integrates stochastic exploration strategies by parameterizing actions as distributions rather than deterministic outputs. Specifically, the policy network outputs both the mean and standard deviation, defining a Gaussian distribution from which actions are sampled as reported in Equation 2.17.

$$\begin{aligned} (\mu_\vartheta, \sigma_\vartheta) &= f_\vartheta(\mathbf{x}_t) \\ \mathbf{u}_t \sim \pi_\vartheta(\mathbf{u}_t|\mathbf{x}_t) &= \mathcal{N}\left(\mu_\vartheta, \mathrm{diag}(\sigma_\vartheta^2)\right) \end{aligned} \tag{2.17}$$

This stochastic approach contrasts with deterministic policies and facilitates more thorough exploration of complex state-action spaces. The critic in SAC evaluates the quality of the actor's selected actions through the TD error, continuously adjusting to reflect the evolving policy. This interaction between actor and critic results in an iterative learning loop, where the actor optimizes actions based on critic evaluations, and the critic updates its estimates based on the actor's behavior, continuing until the desired performance is achieved or external intervention occurs.

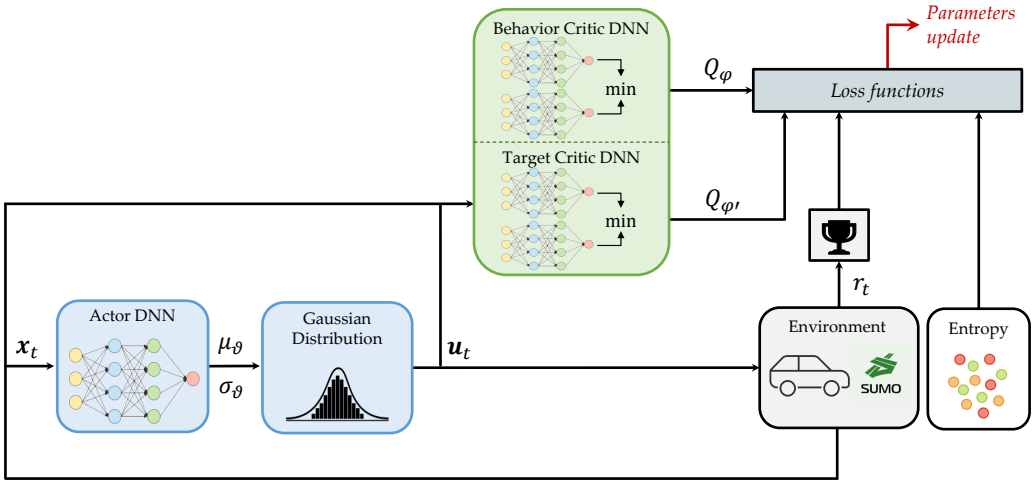


Fig. 2.6 Schematic representation of the architecture of the SAC agent.

The distinctive feature of SAC, entropy maximization, quantifies the randomness in policy actions, calculated via the information entropy:

$$H(\pi(\cdot|\mathbf{x}_t)) = \mathbb{E}_{\mathbf{u}_t \sim \pi}\left[-\log \pi(\mathbf{u}_t|\mathbf{x}_t)\right] \tag{2.18}$$

Higher entropy corresponds to increased randomness, beneficially enhancing exploration and preventing premature convergence to local optima. SAC modifies the conventional Bellman equation (Equation 2.11) by integrating an entropy regularization term, forming the soft Bellman equation:

$$Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = r(\mathbf{x}_t, \mathbf{u}_t) + \gamma \mathbb{E}_{\substack{\mathbf{u}_{t+1} \sim \pi \\ \mathbf{x}_{t+1} \sim P}} \left[Q^{\pi}(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) + \alpha_T H(\pi(\cdot|\mathbf{x}_{t+1})) \right] \tag{2.19}$$

Here, α_T, the entropy regularization coefficient or the temperature parameter, dynamically balances immediate rewards and entropy, directly influencing the exploratory behavior of the policy throughout training. The SAC training is based on an experience replay buffer $\mathscr{D}$, from which it is possible to approximate the Bellman equation through data, by sampling mini-batches of $\mathscr{M}$ experience, like for the DDQL. Thus, the entropy regularized TD target is formulated as follows:

$$\begin{aligned} y(\mathbf{x}_t, \mathbf{u}_t) &= r(\mathbf{x}_t, \mathbf{u}_t) + \gamma \min_{j=1,2} Q_{\varphi',j}(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) - \alpha_T \log \pi_{\vartheta}(\mathbf{u}_{t+1}|\mathbf{x}_{t+1}) \\ \mathbf{u}_{t+1} &\sim \pi_{\vartheta}(\cdot|\mathbf{x}_{t+1}) \end{aligned} \tag{2.20}$$

It should be noted that, with this approach the information entropy $H(\pi(\cdot|\mathbf{x}_{t+1}))$ is approximated by sampling the next action $\mathbf{u}_{t+1}$ from the actual policy π. The SAC algorithm also incorporates the clipped double-Q trick to mitigate Q-value overestimation biases, utilizing two independent critic networks initialized differently. The final loss function guiding the critics updates becomes:

$$J_{Q_{\varphi,j}} = \mathbb{E}_{\{\mathbf{x}_t, \mathbf{u}_t, r_t, \mathbf{x}_{t+1}\} \sim \mathscr{D}} \left[\left(Q_{\varphi,j}(\mathbf{x}_t, \mathbf{u}_t) - y(\mathbf{x}_t, \mathbf{u}_t) \right)^2 \right], \quad j \in 1,2 \tag{2.21}$$

The Q-value estimators' parameters can be updated with a custom frequency or softly updated every training step, like for the DDQL (see Equation 2.16).

The policy network updates aim to maximize the following loss function, which integrates both reward expectations and policy entropy:

$$J_{\pi_\vartheta} = \mathbb{E}_{\{\mathbf{x}_t, \mathbf{u}_t, r_t, \mathbf{x}_{t+1}\} \sim \mathscr{D}} [-\alpha_T \log \pi_\vartheta(\mathbf{u}_t|\mathbf{x}_t) + \min_{j=1,2} Q_{\varphi,j}(\mathbf{x}_t, \mathbf{u}_t)] \tag{2.22}$$

Lastly, the temperature parameter α_T can be kept constant during training or adaptively tuned by minimizing the entropy loss function:

$$J_{\alpha_T} = \mathbb{E}_{\{\mathbf{x}_t, \mathbf{u}_t, r_t, \mathbf{x}_{t+1}\} \sim \mathscr{D}} [-\alpha_T \log \pi_\vartheta(\mathbf{u}_t|\mathbf{x}_t) - \alpha_T \mathscr{H}] \tag{2.23}$$

with $\mathscr{H}$ representing the desired target entropy, providing an explicit goal for policy randomness.

SAC represents a significant advancement over previous methods, addressing both exploration and stability concerns in complex continuous action spaces, solidifying its role as a leading-edge solution for sophisticated RL applications.

Chapter 3

Control Application

The two scales investigated in this dissertation, i.e., city-wide coordination of AMoD fleets and component-level power-split control in hybrid powertrains, share a profound structural connection from the standpoint of systems control. In both domains, energy efficiency is significantly enhanced when intelligent controllers perceive and respond dynamically to real-time conditions. Furthermore, both systems operate within stochastic, highly interconnected environments. At the fleet level, fluctuating traffic conditions continually affect travel times, while at the powertrain level, traffic dynamics influence vehicle speeds imposed by drivers, introducing substantial uncertainty into powertrain load demands. Additionally, each scenario presents a high-dimensional state–action space: thousands of vehicles for fleet management and multiple hybrid operation modes and torque-split configurations at the powertrain level. Consequently, researchers must deal with the curse of dimensionality, inherent stochasticity, and strict computational limitations.

Due to these parallel structures, the evolution of control methodologies in these distinct domains has unfolded along remarkably similar trajectories. Initially, Rule-Based (RB) heuristics dominated, providing rapid decisions but often lacking of optimality. Subsequently, deterministic optimization frameworks emerged, as Mixed-Integer Programming (MIP) for fleet management, and DP and Pontryagin-type methods for powertrain control, offering improved performance but suffering from exponential computational growth and a heavy reliance on prior system knowledge. More recently, DRL introduced a paradigm shift, promising near-optimal decision-making with significantly reduced inference times. Leveraging NN approximators

within AC or value-based approaches, DRL efficiently handles high-dimensional state spaces, identifies near-optimal strategies, and continuously adapts to dynamic and uncertain environments.

This dissertation capitalizes on these converging insights to craft a unified RL framework capable of spanning both spatial and temporal scales, from city-wide mobility systems to hybrid powertrains. In what follows, the relevant literature is synthesized into a cohesive narrative, underscoring shared technical challenges and illustrating how cross-domain insights between fleet logistics and vehicular energy management enrich the controller design proposed herein.

AMoD coordination

In AMoD systems, a primary challenge arises from the inherently high-dimensional state–action space, naturally framed as a networked optimal control problem. Extensive research across various domains, including power systems [71], telecommunications [72], supply-chain logistics [73], and transportation [74, 1], has demonstrated that large-scale operational tasks can often be recast as graph-based flow optimization problems. Traditional tools, such as time-expanded networks [75] and Linear Programming (LP) [76], provide optimal solutions under deterministic conditions but struggle significantly in real-world scenarios where stochasticity undermines predictability and computational burdens escalate exponentially.

Reflecting this broader progression, early AMoD operators relied on fast but short-sighted heuristics [77, 78], trading efficiency for computational simplicity. Advances in MPC and mixed-integer optimization significantly boosted system performance by re-optimizing decisions over finite horizons [79, 80]. Nevertheless, these methods exhibit super-linear runtime growth relative to fleet size and degrade under unpredictable conditions like sudden demand changes or traffic disturbances. Recent trends thus pivoted toward data-driven and RL-based methods that derive policies through interaction with simulated mobility environments. Notable contributions include AC policies with fleet-size-independent action spaces [81], sequential assignment strategies with theoretical performance guarantees [82], and decentralized multi-agent controllers tailored for electric fleets [83]. Yet, many of these methods do not exploit the inherent structure of the transportation network. Addressing this gap, Gammelli et al. [1, 53] incorporated GCN into hierarchical policies, effectively reducing dimensionality and exploit information from neigh-

boring areas in the city to further optimize the decisio-making. Building upon this, Schmidt et al. [26] and Gammelli et al. [54] further enhanced sample efficiency and robustness through offline RL and meta-RL training paradigms [84]. Despite significant advances, however, policies are predominantly validated using simplified, high-speed simulators, leaving unresolved issues regarding their robustness and applicability in noisy real-world environments.

Energy Management System for Hybrid Powertrains

The evolution of EMS for hybrid powertrains mirrors the trajectory observed in AMoD coordination. Industry widely employs RB strategies, such as lookup tables and threshold-based logic [85], due to their transparency and minimal computational requirement. Nevertheless, RB strategies embed expert biases and often perform very sub-optimally outside their calibration scenarios, limiting their effectiveness under diverse real-world conditions. In pursuit of optimality, research has explored global optimization methods such as DP [69] and Pontryagin's Minimum Principle (PMP) [86], offering optimal solutions if the complete mission profiles are known in advance. However, these methods are computationally intensive and impractical for real-time deployment. Alternative local optimization techniques, including ECMS [87] and MPC [88], mitigate computational costs but risk suboptimal performance, particularly under cycle variations [89].

The advent of RL provided an alternative and efficient solution, progressing from tabular QL, which is effective but limited by state-action space discretization [90–92], to DQL, which leverages NN value approximators to manage continuous states more efficiently [93–95]. Despite these improvements, DQL remains susceptible to action discretization errors and overestimation bias. AC methods, such as DDPG and Twin Delayed Deep Deterministic Policy Gradient (TD3) [96], successfully addressed these limitations, finding application in multi-objective EMS, battery health management, and waste-heat recovery scenarios [97–100]. Recently, entropy-regularized SAC algorithms have emerged prominently, offering superior exploration–exploitation balance and training stability [101–104].

Yet, persistent challenges parallel those in AMoD research: most EMS studies risk overfitting by training policies on singular driving cycles, and simplistic vehicle models frequently hinder effective transition to detailed virtual vehicle test rigs and real-world test cases.

Converging Themes and Remaining Challenges

Despite their distinct operational scales, AMoD fleet coordination and hybrid powertrain EMS share a common set of scientific challenges:

- navigating vast, continuous state–action spaces under real-time computational constraints;
- maintaining robustness among stochastic, dynamic traffic conditions, unpredictable demand, and varied driver behavior;
- leveraging structural domain knowledge (network topology or drivetrain physics) to mitigate the curse of dimensionality;
- ensuring sample-efficient, safe training procedures, given the constraints of computationally intensive simulations and real-world safety demands;
- achieving generalization across varying geographic locations, temporal scales, and operational contexts, bridging the persistent "sim-to-real" gap.

Contributions

This dissertation harmonizes two distinct yet structurally similar research streams: city-scale AMoD coordination and component-level hybrid-powertrain EMS optimization. The contributions build upon and extend previous results reported in [105] for fleet coordination, and in [38, 95, 104, 2, 106] for the powertrain domain. Specifically:

- At the AMoD scale, a SAC framework is adopted, integrating a three-layer hierarchical policy proposed by [1] to significantly compress the action space. The policy controls idle vehicle redistribution across an aggregated city network, substantially enhancing computational tractability. Training and evaluation conducted in a SUMO-based mesoscopic traffic simulator [55] demonstrate impressive robustness and generalization across various conditions, including different network aggregation levels, simulator fidelities, and time-of-day variations.

- At the powertrain scale, the dissertation applies SAC initially on a simplified vehicle model, subsequently validating the learned policies on a high-fidelity virtual test rig using a Real Driving Emissions (RDE) compliant cycle. Extensive cross-validation confirms robust performance across diverse driving missions and initial battery energy conditions, verifying the scalability and practical applicability of the developed approach from simplified training conditions to realistic operational environments.

Collectively, these contributions demonstrate the feasibility and effectiveness of using a unified DRL framework, specifically SAC, across multiple scales and domains, thus addressing significant real-world challenges in transportation efficiency and sustainability.

Chapter Structure

This chapter is structured to reflect the two distinct yet interconnected scales while maintaining a cohesive narrative flow from modeling foundations to control design and performance evaluation. Section 3.1 establishes the foundational modeling frameworks: Section 3.1.1 analyzes various traffic simulator paradigms (macroscopic, mesoscopic, and microscopic) and motivates the selection of SUMO's mesoscopic engine for fleet-level analyses, while Section 3.1.2 details the modeling strategies employed for vehicle longitudinal dynamics and hybrid powertrain behavior. Building upon these modeling foundations, Section 3.2 introduces a unified optimal control perspective, critically reviews existing optimization-based methodologies, contrasts them with RB heuristics, and provides a rationale for adopting learning-based approaches, particularly the SAC framework. Subsequently, the chapter is split to address each application scale separately yet consistently. Section 3.3 details the methodology, presents a specific case study, and discusses the principal results concerning fleet coordination. In parallel, Section 3.4 mirrors this structure at the component level, focusing on powertrain EMS optimization and associated outcomes.

3.1 Modeling Theory

A rigorous evaluation of control policies for AMoD fleets and hybrid-electric powertrains depends on equally rigorous system-level models. At the network scale, traffic simulators translate origin–destination demand into time-resolved vehicle trajectories, exposing the spatial–temporal patterns that rebalancing and charging algorithms must master. Microscopic platforms track each individual car, macroscopic tools approximate continuous traffic streams, and mesoscopic engines lie in between; selecting the right fidelity, therefore governs both realism and computational cost for AMoD studies. At the vehicle scale, electrified–powertrain models convert those trajectories into fuel and electricity use, capturing how torque is shared between engine, motor, and battery under a chosen EMS. Together, traffic and xEV models form a digital twin that connects city-wide flows with component-level energy balances.

The remainder of this chapter reflects that dual perspective. Section 3.1.1 surveys traffic-simulation paradigms in ascending order of resolution, macroscopic, microscopic and mesoscopic, highlighting their core assumptions, typical use-cases and relevance to fleet coordination. Section 3.1.2 then offers a concise introduction to electrified-vehicle classes before detailing the modeling stack adopted in this thesis: longitudinal vehicle dynamics, steady-state map representations of engine, motor, and battery, and the backward- and forward-quasi-static numerical schemes used to propagate those models along drive cycles. Each subsection establishes the assumptions, parameters, and solution methods later employed in the AMoD and EMS case studies.

3.1.1 Traffic Simulators

Accurate traffic modeling is the foundation of any realistic simulation of AMoD systems, because the travel times, network congestion levels, and vehicle to vehicle interactions predicted by the traffic layer feed directly into vehicle routing, fleet sizing, energy and emission assessments of the AMoD fleet. No single traffic simulator is universally “best”: instead, researchers choose among three classical categories: macroscopic, mesoscopic, and microscopic, by weighing a fundamental trade-off simulator fidelity and computational cost. The following subsections detail (i) the continuum and graph-based foundations of macroscopic models, (ii) the queue-based mesoscopic formulation implemented in SUMO, and (iii) the vehicle-dynamics ingre-

dients of microscopic simulation, highlighting how each layer supports progressively richer, but costlier, analyses of AMoD operations.

Macroscopic

Macroscopic models treat traffic flow as fluid flow on a road network graph, focusing on aggregate quantities like flow, density, and speed. The road network is represented as a graph with nodes that represent the intersections, connected by edges that represent the road segments. The traffic dynamics on each node are described by continuum equations, whose main equation is the conservation of vehicles on each node:

$$\begin{cases} \frac{\partial \rho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0 \\ q(x,t) = Q\left(\rho(x,t)\right) \end{cases} \tag{3.1}$$

where $\rho(x,t)$ is the traffic density (measured in $[veh/m]$, describing the mean number of vehicles per unit roadway length, while $q(x,t)$ is the traffic volume or flow rate, describing the mean number of vehicles passing the cross-section x per unit time at time t (measured in veh/s). This equation states that changes in traffic density $\rho(x,t)$ along a road are due to imbalance of inflow $q(x,t)$ and outflow. This equation, according to the Kinematic Wave Model [107], is paired with the fundamental diagram Q that relates the flow $q(x,t)$ to the density $\rho(x,t)$ and thus to the traffic speed along the road through $q(x,t) = \rho(x,t)\dot{x}$. A classic example is the Lighthill-Whitham-Richards (LWR) model [108, 109], a first-order model where each road behaves like a single homogeneous stream with a velocity that decreases with density. This continuum approach captures phenomena like shockwaves and traffic jams as wave propagations in the density field. Graph-based approaches usually discretize the continuum model for computation. A notable method is the Cell Transmission Model (CTM) proposed by Daganzo in 1994 [110]. It divides each road intersection into cells and updates the vehicle count in each cell at small time steps. The update rule for each cell i is given by:

$$n^i_{t+\Delta t} = n^i_t q^i_t - q^{i+1}_t \tag{3.2}$$

where n_t^i is the number of vehicles in cell i and q_t^i is the flow, so vehicles leaving cell i, during the interval $(t, t+\Delta t)$. The flow q_t^i is determined by upstream demand and downstream supply, ensuring that it does not exceed the road's capacity or cause negative densities. In essence, each link acts like a pipeline with a finite wave propagation speed, and vehicles entering a link will propagate cell by cell unless impeded by downstream congestion. These macroscopic network models are computationally efficient and well-suited for large-scale simulations. They can reproduce key emergent phenomena, like formation and dissipation of queues, shockwave propagation, and are deterministic, avoiding the noise of individual vehicle behavior. An added benefit from an energy perspective is that macroscopic models readily provide aggregate measures like average travel time, fuel consumption, and emissions for the network. For example, by using aggregate speed and flow outputs, one can estimate fuel usage or emission rates with appropriate consumption models. This makes macroscopic graph-based simulators attractive in transport-energy studies where system-wide efficiency or emissions are evaluated.

Mesoscopic

Mesoscopic traffic models combine the aggregate efficiency of macroscopic frameworks with the detailed vehicle representation of microscopic simulations, relying on simplified vehicle dynamics derived from queueing theory [111]. In these models, each road link is split into *queues* with finite capacity, where vehicles travel at free-flow speed until they reach the end of the segment and may join a FIFO queue if the discharge capacity is saturated. The queue length $N(t)$ evolves according to the accumulation equation:

$$\dot{N}(t) = q_{\text{in}}(t) - \mu, \tag{3.3}$$

where $q_{\text{in}}(t)$ is the arrival rate and μ is the link's outflow capacity. In this work, the mesoscopic engine of SUMO [55] is employed to simulate traffic dynamics for AMoD systems efficiently. Each network k-th edge is represented as a single aggregated queue of fixed length L_k, refreshed in discrete steps Δt, with each vehicle v assigned an exit time t_{exit}^v computed as:

$$t_{\text{ff}}^{v} = \frac{L_k}{\min\{v_{\text{max}}^{v},\, v_{\text{lim}}^{k}\}} \tag{3.4}$$

$$t_{\text{exit}}^{v} = \max\left\{t_{\text{in}}^{v} + t_{\text{ff}}^{v},\, t_{\text{exit}}^{v-1} + H_s^k(o^k, o_{\text{jam}})\right\} \tag{3.5}$$

where v_{max}^{v} is the desired speed of vehicle v, v_{lim}^{k} is the speed limit, and H_s^k is the minimum time headway ensuring safe progression based on the congestion state of the current and downstream queues. Congestion is assessed through the occupancy $o^k = n^k L_{\text{veh}}/L_k$, with n^k being the number of vehicles in the queue and o_{jam} a jam threshold, which must be properly calibrated to ensure realistic congestion through the network. Four headway regimes are defined as:

$$H_s^k = \begin{cases} H_{\text{ff}}, & o^k < o_{\text{jam}},\, o^{k+1} < o_{\text{jam}}, \\ H_{\text{fj}}, & o^k < o_{\text{jam}},\, o^{k+1} \geq o_{\text{jam}}, \\ H_{\text{jf}}, & o^k \geq o_{\text{jam}},\, o^{k+1} < o_{\text{jam}}, \\ H_{\text{jj}}, & o^k \geq o_{\text{jam}},\, o^{k+1} \geq o_{\text{jam}}, \end{cases} \tag{3.6}$$

capturing both free-flow and congested interactions. Intersections are modeled as additional delays based on traffic lights or priority rules. By abstracting continuous vehicle trajectories into queue-based updates, SUMO's mesoscopic solver is up to two orders of magnitude faster than its microscopic engine, yet still reproduces congestion effects and realistic travel times, making it ideal for AMoD simulations and energy analysis at the fleet level.

Microscopic

Microscopic models simulate the detailed behavior of individual vehicles. Each vehicle's acceleration and spacing are computed based on car-following models, for the longitudinal motion, and lane-changing rules, for lateral movement. In the open-source simulator SUMO, several car-following models are implemented to govern how a vehicle follows the one in front of it. These models are defined by equations that ensure collision-free, yet realistic, interactions between vehicles. The default car-following model in SUMO is the Krauss model [112] which is essentially a stochastic safe-distance model. It can be considered a space-continuous version of

the Nagel-Schreckenberg cellular automaton model [113]. For each follower vehicle v, the maximum safe speed is determined by the braking-safety inequality:

$$v_{\text{safe}} \leq v^{\text{lead}} + 2\,b_{\max}\left(g - \frac{v\,v^{\text{lead}}}{2\,b_{\max}}\right)^{1/2} \tag{3.7}$$

where g is the net gap, v^{lead} the leader's speed and $b_{\max}$ the (common) comfortable deceleration; the adopted speed is then:

$$v_{t+\Delta t} = \max\{0,\,(1-\omega)\,v_{\text{safe}} + \omega\,\xi\} \tag{3.8}$$

with $\omega \in [0,1]$ a driver imperfection factor and ξ a uniform random draw, guaranteeing collision-free operation by construction. The widely used Intelligent Driver Model (IDM) [114] is also implemented and often preferred for smooth trajectory analysis. Its acceleration law:

$$\begin{cases} \dot{v} = a_{\max}\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v,\Delta v)}{s}\right)^2\right] \\ s^* = s_0 + v\,H_s + \frac{v\,\Delta v}{2\sqrt{a_{\max} b}} \end{cases} \tag{3.9}$$

depends on desired speed v_0, time headway H_s, minimum jam gap s_0, maximum acceleration $a_{\max}$, comfortable deceleration b, and exponent δ , producing realistic acceleration and braking profiles that are crucial for emission and energy studies at the vehicle level. Lane selection is handled by the MOBIL rule [115], which evaluates a prospective lane change by the criterion:

$$a_i^{\text{new}} - a_i^{\text{old}} + p\left(a_\ell^{\text{new}} - a_\ell^{\text{old}} + a_{\ell-1}^{\text{new}} - a_{\ell-1}^{\text{old}}\right) > \Delta a_{\text{thr}} \tag{3.10}$$

with p the driver politeness factor and Δa_{thr} a minimum incentive threshold. Signalised junctions are simulated by a discrete phase optimizer that imposes amber- and red-light constraints on incoming links; unsignalised and priority nodes follow gap-acceptance logic that yields realistic blocking . Microscopic simulation therefore captures stop–and–go waves, capacity drop at merges, and the dynamic effect of Adaptive Cruise Control (ACC) or eco-driving algorithms, features essential when the research goal is to relate instantaneous powertrain behaviour to traffic conditions. The cost is computational: memory scales with the number of vehicles and runtime grows

linearly with $N_{\text{veh}}/\Delta t$, limiting network size or horizon length unless multi-core processing and adaptive time stepping are used. Nevertheless, for corridor studies or the calibration of powertrain energy models, the microscopic layer provides the necessary fidelity that coarser mesoscopic or macroscopic abstractions cannot.

3.1.2 xEV modeling

Traffic models describe how individual vehicles interact on the road network, yielding speed–position trajectories for every vehicle. Translating those trajectories into reliable estimates of energy use and pollutant emissions, however, requires a description of the vehicle itself and, in particular, of how its powertrain converts traction demands into fuel and electricity consumption. The present work focuses on HEVs and on assessing innovative EMSs for such architectures. To that end, the vehicle model must satisfy two often conflicting requirements:

- *Physical fidelity*: it must capture the dynamics of the powertrain components, i.e., ICE, Electric Motor (EM), battery, and satisfy the traction demand coming from the road and the driver;
- *Computational efficiency*: it must run fast enough to process long drive cycles, and optimization loops within a reasonable computational time.

These considerations motivate the adoption of a system-level modeling framework, which is detailed in this section. The formulation begins with a brief introduction of the electrified powertrain in Section 3.1.2, followed by the description of the longitudinal dynamics of the vehicle, presented in Section 3.1.2, and the modeling approach for the hybrid powertrain components in Section 3.1.2. Finally, Section 3.1.2 outlines the numerical methods used to solve the coupled dynamics of the vehicle and powertrain efficiently and accurately.

Introduction to Electrified Vehicles

Electrified Vehicles (xEVs) involve a some technologies that use electricity as a primary or auxiliary source for vehicle propulsion, replacing conventional energy sources and associated components with their electric counterparts. Generally, it's

possible to electrify not only the propulsion system, but also some auxiliary systems that traditionally use energy coming from the ICE, degrading its overall efficiency. BEVs use electric motors powered by batteries instead of ICE, but their market spread is limited by range, charging time, and infrastructure challenges [16]. In between traditional ICE powertrains and BEVs, HEVs represent a compromise by integrating the advantageous aspects of both propulsion solutions, thus, it's considered a promising solution at least in the mid-term scenario. The high energy density of fossil fuels enables the ICE to provide extended driving range, while the addition of an auxiliary energy storage system allows the load to be shared between the ICE and the EM. This sharing can move the ICE operating points into more efficient or less polluting regimes. Furthermore, in HEVs, the ICE can be deactivated when not needed and, compared to conventional vehicles, the ICE can be downsized and paired with EMs, resulting in improved fuel economy without affecting the overall performance of the powertrain. Additionally, kinetic energy lost during braking can be partially recuperated, as EMs function as generators during braking events, converting kinetic energy into electrical energy to be stored in the battery, enabling the so-called regenerative braking. Among the xEV powertrains, it's possible to further categorize them depending on the powertrain electrification level. Figure 3.1 presents a classification of xEVs based on the relative size of their electric actuators. The higher the EM and battery sizes, the higher the degree of electrification. In conventional vehicles, the ICE alone fulfills all propulsion needs. Micro-hybrids add a small EM, enabling start-stop functionality to reduce idling fuel use and emissions. Mild and full hybrids use larger EMs and batteries, providing additional features such as engine assistance and regenerative braking. Plug-in Hybrid Electric Vehicles (pHEVs) differ from HEVs in that their larger batteries can be recharged externally, enabling longer travel distances with minimal ICE intervention. At the far end of the spectrum, fully electric vehicles rely exclusively on EMs, which may be powered by batteries or fuel cells.

HEVs can also be categorized according to their powertrain architecture, typically falling into one of three main types:

- Series Hybrid: only the EMs are mechanically linked to the wheels. Traction is provided by EMs powered either directly from the battery or by a generator driven by the ICE.

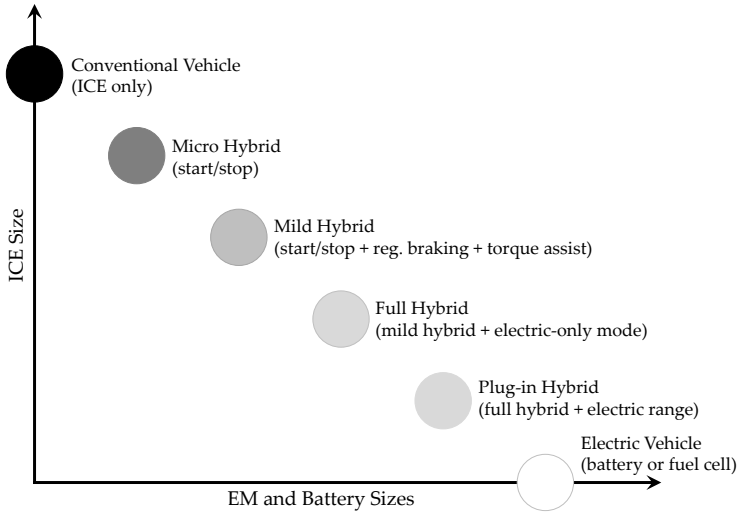


Fig. 3.1 Spectrum of vehicle electrification levels.

- Parallel Hybrid: both the ICE and the EM can supply their mechanical power to the driven wheels, and the connection between the ICE and the EM is obtained by means of a mechanical transmission system;
- Complex Hybrid: this architecture is characterized by the coexistence of a parallel path with a series one.

Vehicle modeling

With vehicle's longitudinal dynamics analysis, the vehicle can be treated as a single lumped mass. Applying Newton's second law to this point-mass representation (see Figure 3.2) give:

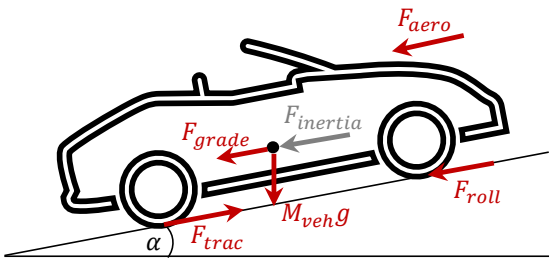


Fig. 3.2 Forces acting on a vehicle.

$$M_{veh}\dot{v} = F_{inertia} \tag{3.11}$$

where M_{veh} denotes the total mass of the vehicle, including the equivalent inertias of all driveline components, v is the longitudinal velocity, and $F_{inertia}$ is the inertial force. That inertial component can be decomposed as:

$$F_{inertia} = F_{pwt} - F_{brk} - F_{roll} - F_{aero} - F_{grade} \tag{3.12}$$

with F_{pwt} as the tractive force produced by the powertrain, F_{brk} as the resistive force applied by the braking system, F_{roll} as the rolling resistance, F_{aero} as the aerodynamic drag, and F_{grade} gravitational component coming from road slope. Usually, the rolling resistance can be commonly expressed as [17]:

$$F_{roll} = c_{roll} M_{veh} g \cos\alpha \tag{3.13}$$

where g is the gravitational acceleration, α is the road slope angle, and c_{roll} is the rolling resistance coefficient. The latter depends on speed, tyre inflation pressure, and ambient temperature, and is often approximated by a polynomial expression depending on the vehicle longitudinal velocity:

$$c_{roll} = c_0 + c_1 v + c_2 v^2 + c_3 v^3 \tag{3.14}$$

The aerodynamic resistance is calculated via:

$$F_{aero} = \frac{1}{2}\rho_{air} A_f C_d v^2 \tag{3.15}$$

where ρ_{air} is the air density, A_f is the vehicle frontal area, and C_d is the drag coefficient. Finally, the component of gravitational force along the road incline is:

$$F_{grade} = M_{veh} g \sin\alpha \tag{3.16}$$

However, since isolating aerodynamic drag from rolling resistance during testing is often difficult, both effects are commonly experimentally determined, through a procedure known as coast-down test: i.e., the vehicle is allowed to decelerate freely,

so that the observed deceleration arises exclusively from these two resistances; by recording the instantaneous speed, one can then compute the total drag force acting on the vehicle as a function of velocity:

$$F_{roll+aero} = C_0 + C_1 v + C_2 v^2 \tag{3.17}$$

where C_0, C_1, and C_2 are the Coast-Down coefficients [116].

These relations form the fundamental framework for longitudinal vehicle modeling; with accurately identified parameters, they deliver an accurate representation of the vehicle response for most powertrain and energy analyses.

Powertrain modeling

Building on the longitudinal dynamics formulated in Section 3.1.2, the powertrain model considered here contains the main components common to most hybrid architectures, an ICE, an EM with its power-electronics stage, and a battery pack, that act together to meet the wheel torque demand, as reported in Figure 3.3. The actual number of components and the level of modeling detail can vary with the specific hybrid configuration (series, parallel, power-split, etc.); for clarity, only these principal subsystems are described in the following text.

Internal Combustion Engine. As illustrated in Figure 3.4, engine-modeling techniques range from full 3D-CFD, which models in-cylinder combustion and gas-exchange physics at a very high computational cost, through 1D-CFD, which solves mono-dimensional conservation equations for the entire intake–exhaust network with moderate run-times, down to map-based models that models the engine behavior with steady-state lookup tables of fuel flow, efficiency, and emissions versus speed and load. The map-based approach can run entire driving missions in, or faster than, real time, making it the method of choice for energy-related analysis at the vehicle system level. However, their lumped nature means that accuracy degrades whenever rapid transients, turbo-lag, or other dynamic gas-exchange effects become central to the investigation.

Electric Machine. At the system level, like an ICE is described through steady-state performance maps, an EM is modeled with look-up tables. Due to the machine's very high rotational speeds, the rotor's lumped inertia is the only dynamic state

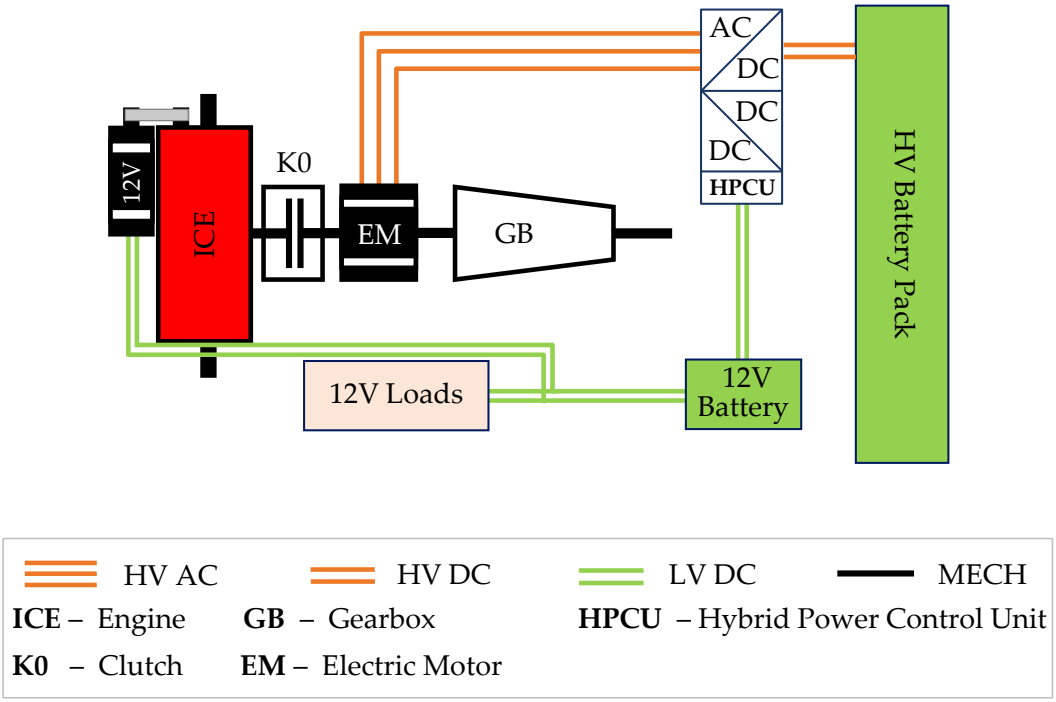


Fig. 3.3 Example of a hybrid powertrain with its main components.

explicitly simulated, while the much faster electromagnetic dynamics are embedded into the map representation. Depending on the chosen formulation, the electrical and mechanical power/torque are coupled with an efficiency map expressed versus speed and load. Losses in the power-electronics stage can be modeled through a dedicated efficiency map or directly embedded into the EM's overall efficiency map. In traction mode, the mechanical power at the shaft P_{mech} is obtained from the electrical power P_{elec} through:

$$P_{mech} = \frac{P_{elec}}{\eta} \tag{3.18}$$

whereas in generator mode, the relationship reverses to:

$$P_{mech} = P_{elec}\eta \tag{3.19}$$

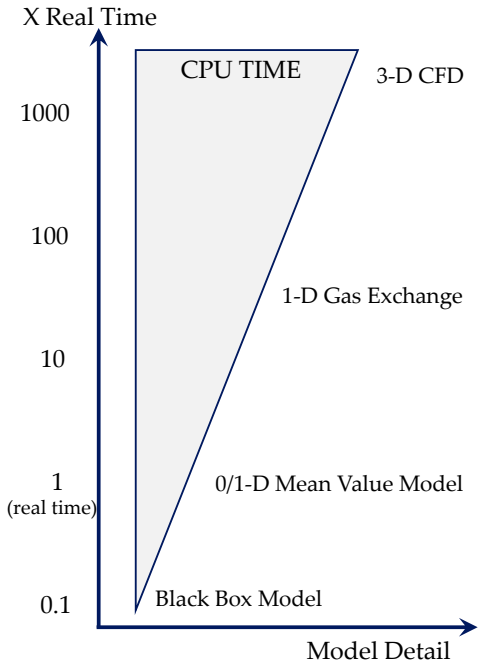


Fig. 3.4 Main engine modeling methodologies: model detail vs. computational time.

The power sign distinguishes traction from regeneration, whose sign depends on the convention used, but the losses are always positive and are computed as:

$$P_{loss} = P_{elec} - P_{mech} \tag{3.20}$$

Battery. As with ICEs and EMs, batteries can be represented with models of varying fidelity, each featuring the usual trade-off between accuracy of the captured physics against computational cost. At the highest resolutions are atomic-scale simulations, used for material behavior characterization, molecular-scale models that solve phase transitions within cell constituents, and microscale frameworks based on non-equilibrium thermodynamics that recreate the 3D morphology of porous electrodes. For vehicle-level studies, however, it is common to describe the pack's electrical behavior with Equivalent Circuit Models (ECMs), which provide an efficient yet sufficiently accurate representation of battery electrical dynamics for driving cycle simulations and control development. Following the system-level models approach already adopted for the ICE and EM subsystems, the discussion that follows is restricted to the ECM description.

Figure 3.5 shows a typical ECM for a battery. In this representation, the resistor R_0 accounts for ohmic losses due to wire resistance, electrode resistance, and other dissipative effects. Dynamic behavior is captured by a parallel resistor–capacitor pair, R_1 and C_1. Although the illustration uses a single R-C branch (first-order model), additional branches can be added to improve fidelity.

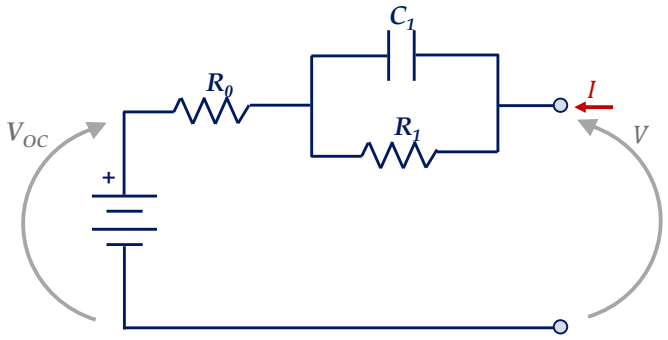


Fig. 3.5 Battery equivalent circuit-based model (first-order).

The terminal voltage is given by

$$V_L = V_{oc} - R_0 I - \sum_{i=1}^{n} V_i \tag{3.21}$$

where V_{oc} is the Open Circuit Voltage (OCV) measured after the cell has reached equilibrium under unloaded conditions, I is the current, and V_i denotes the voltage across the ith R-C branch. Here n is the number of such branches ($n = 1$ in the figure). Each branch is described by the first-order differential relation

$$C_i \frac{dV_i}{dt} = I - \frac{V_i}{R_i} \tag{3.22}$$

Both R_i and C_i may vary with operating conditions such as temperature and the sign of the current (charging vs. discharging). Their numerical values, along with R_0, are usually obtained by fitting the model to experimental data, with the complexity of the task increasing with the number of R-C branches.

A still simpler representation replaces the entire R-C network with one lumped internal resistance, R_int, in series with the open-circuit voltage source. All ohmic and polarization effects are lumped into this single parameter, reducing the terminal-voltage relation to

$$V_L = V_{oc} - R_{int} I \tag{3.23}$$

As with the individual R-C elements, $R_i nt$ is identified through curve-fitting of experimental data and is usually expressed as a function of operating parameters, most commonly the cell State of Charge (SoC) and temperature, and, where necessary, current direction.

Numerical Models

Given the equations for the vehicle and hybrid powertrain models, they must be embedded in a computational framework that predicts energy use and fuel consumption over a specified drive cycle. Two quasi-static approaches are commonly use in practice.

Backward Kinematic Approach. The backward approach is commonly employed to estimate a vehicle's fuel consumption and pollutant output over a driving mission. In this framework, the speed profile and road gradient are imposed a priori, and the calculation proceeds backward from the wheels to the energy sources, without resolving the internal dynamics of the powertrain components. Each machine is therefore represented by steady-state look-up maps (see Section 3.1.2), a choice that delivers high computational speed and yields overall fuel and emissions output that are sufficiently accurate for preliminary assessments over the driving cycle. The cost of this efficiency is that transient phenomena are reduced to a sequence of steady-state operating points, so results can change from experimental data during highly dynamic events. The information flow of the backward method is illustrated in Figure 3.6.

Fig. 3.6 Information flow in a backward kinematic approach.

Quasi-static Forward Approach. In a quasi-static forward simulation, the ICE, EM, and battery are still represented by steady-state look-up tables; however, unlike the backward kinematic method, vehicle speed is not imposed on the vehicle anymore, but treated as a target profile. A driver model, typically implemented as a Proportional-Integral-Derivative (PID) controller, modulates brake and throttle commands so that the actual speed follows the target. The longitudinal dynamics equation then updates acceleration and velocity at each time step, thereby capturing the overall system dynamics that the backward approach lumps into the equivalent vehicle mass (see 3.1.2. The corresponding flow of information is reported in Figure 3.7.

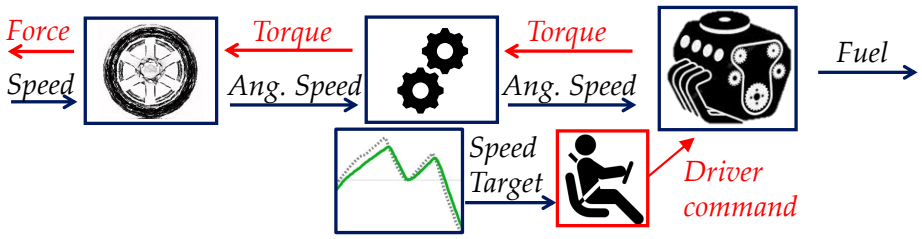


Fig. 3.7 Information flow in a forward dynamic approach.

Although this quasi-static scheme delivers reasonably accurate fuel consumption estimates over driving cycles, its steady-state maps cannot resolve strongly nonlinear phenomena such as soot formation, rapid load transients, or turbocharger lag. Analyses that require such detail demand higher fidelity engine models, e.g., 0-D or 1-D fluid-dynamic formulations, to reproduce the full thermodynamic and gas-exchange dynamics.

3.2 Control Theory Background

Efficient management of AMoD fleets and HEV powertrains claims two versions of the same challenge: solving a complex, partially uncertain dynamical system toward long-term objectives while respecting stringent physical and operational constraints. In a city-scale AMoD network, control decisions determine when and where thousands of autonomous vehicles reposition, charge, and serve passengers, directly affecting passenger wait times, operating cost, and overall fleet emissions. At the powertrain scale, EMS for HEVs continuously allocate torque between ICE and

EM to minimize fuel use and pollutants under different battery depletion strategies. Both problems demand real time, foresighted choices, yet being different in their dynamics, horizons, and dimensionality; a unified control theory perspective helps compare solution methods and identify trade-offs in optimality, robustness, and computational burden.

The remainder of this section is organized accordingly. Section 3.2.1 introduces the generic optimal-control formulation used throughout the thesis. Section 3.2.2 reviews optimization-based strategies, proceeding from global DP and PMP to receding-horizon MPC and the instantaneous ECMS variant popular in HEVs. Section 3.2.3 presents heuristics that sacrifice formal optimality for transparency and very low on-board cost, both for fleet rebalancing rules and classical EMS look-up strategies. Finally, Section 3.2.4 motivates data-driven and RL controllers, bridging the gap between model-heavy optimization and model-free heuristics and setting the stage for the case studies presented in later chapters.

3.2.1 Optimal Control Framework

Many engineering and economic tasks can be formulated as optimal control problems: the control theory branch that seeks a control trajectory that guides a dynamical system while maximizing or minimizing a chosen performance index. In the context of HEVs, for example, the control variable might be the instantaneous power-split between the powertrain machines, and the objective could be minimizing fuel consumption, pollutant emissions, or a weighted combination of both. At a larger scale, the motion and spatial distribution of a robotaxi fleet may be coordinated to maximise service profit or minimise fleet-related pollutants or CO_2 emissions, again through an appropriately defined multi-objective index. Classical optimal control methods are feasible only when the underlying mathematical model is simple, due to the computational burden associated with the optimization process itself, and the future evolution of the system is perfectly known over the optimization horizon. Real systems rarely meet these conditions, since model uncertainty and unknown boundary conditions are always present, so any practical implementation is, by construction, only near-optimal. Following the formalism of [117], the system dynamics can be expressed as

$$\dot{\mathbf{x}}(t) = f\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \tag{3.24}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the set of states, $\mathbf{u}(t) \in \mathbb{R}^m$ is the set of controls, f is a function that model their time evolution, and t is the time variable. In an HEV, f includes the vehicle and powertrain equations (see Section 3.1.2; in a fleet model, it may encode vehicle motion through a road network and traffic evolution (see Section 3.1.1). For a problem with fixed final time t_f and partially specified terminal state, the control law $\mathbf{u}(t)$ is optimal over $t \in [t_0, t_f]$ if it minimizes the cost functional

$$J = \phi\left(\mathbf{x}(t_f), t_f\right) + \int_{t_0}^{t_f} L\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt \tag{3.25}$$

subject to the terminal condition (boundary conditions on states)

$$\psi\left(\mathbf{x}(t_f), t_f\right) = 0 \tag{3.26}$$

and instantaneous constraints

$$\begin{cases} G\left(\mathbf{x}(t), t\right) \leq 0 \\ \mathbf{x}(t) \in \mathscr{X}(t) \\ \mathbf{u}(t) \in \mathscr{U}(t) \end{cases} \qquad \forall t \in [t_0, t_f] \tag{3.27}$$

Here, $L\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \in \mathbb{R}$ denotes the instantaneous cost-to-go function, that can be the fuel rate, pollutant emission rate or any other quantities related to a desired dynamic behavior for a HEV, or the monetary profit, and CO_2 or pollutant emissions at the fleet level. Φ captures any terminal penalty, such as final battery SoC, fleet distribution at the end of the day. Additional local constraints, such as physical limitations of the powertrain components, i.e., ICE, EM, battery, or flow conservation at network nodes, restrict admissible trajectories, while integral or terminal constraints enforce conditions at the final time instant t_f. These may be imposed as hard constraints through (3.26) or as soft constraints by augmenting the terminal cost Φ. For instance, if J minimized only instantaneous fuel flow, the optimal solution would exploit a pure electric drive, depleting the battery until the end of the driving mission; incorporating a SoC penalty in Φ prevents this unrealistic outcome and ensures a desired final electric energy content. To solve the optimal

control problem, several families of methods can be employed, depending on how much system knowledge is available, how much detail is required, and whether real-time implementation is needed. *Optimization methods* formulate the task explicitly as an optimization problem, computing the control sequence either instantaneously, over a receding finite horizon, or across the entire time period through global techniques. A second category relies on *RB heuristics*, by carefully tuning decision rules that sacrifice strict optimality in exchange for very low computational cost while still tracking the optimal policy reasonably well. A third, *learning-based*, approach trains a parametrized policy, often using machine learning models, so that after an offline training phase, it can deliver near-optimal performance with the computational speed required for online deployment. The next sections discuss these three classes of techniques in detail.

3.2.2 Optimization-based Approaches

Optimization-based approaches tackle the optimal-control problem by applying mathematical programming techniques that can be grouped into two broad families:

- *Global optimization methods*: the full problem is solved in a single batch under the assumption that past, present, and future operating conditions are perfectly known. Representative algorithms are DP [69] and PMP [86].
- *Local optimization methods*: the overall task is decomposed into a succession of smaller problems. Only current and, when available, predicted near-future information is used, typically within a receding-horizon framework. MPC [88] and Stochastic Dynamic Programming (SDP) [118] fall into this category, while the Equivalent Consumption Minimization Strategy (ECMS) [87], widely adopted for HEVs, performs an instantaneous minimization of its cost function.

Dynamic Programming

DP [69] [117] provides a numerical method for obtaining the globally optimal solution of an optimal control problem. Its main drawbacks are its non-causal nature, since the entire time history of exogenous inputs must be known in advance to

compute the system dynamics across the entire time horizon, and the well-known curse of dimensionality, where computation time increases exponentially with the number of state and control variables. For continuous systems, both the state and control spaces need to be discretized; analytic expressions for the cost-to-go function can be found only in special cases, so spaces discretization is the common approach, even if it introduces numerical errors, which could degrade the accuracy of the solution [119]. DP is based on Bellman's principle of optimality, which states that:

> *"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [69].*

Therefore, the portion of the optimal trajectory that starts at any intermediate stage and continues up to the end must coincide with the corresponding tail of the global optimal policy. The translation of this statement into mathematical form leads to the (3.28). Because DP works in a discrete fashion, time, state, and control variables are first sampled on finite grids. Consider, therefore, a discrete-time system with a fixed terminal instant:

$$\mathbf{x}_{k+1} = f_k(\mathbf{x}_k, \mathbf{u}_k) \tag{3.28}$$

with $k = 0, 1, \ldots, N-1$, $\mathbf{u}_k \in \mathscr{U}_k$, and $\mathbf{x}_k \in \mathscr{X}_k$. A control policy is the sequence:

$$\pi = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}\} \tag{3.29}$$

Its cumulative cost, from step 0 in the initial state $\mathbf{x}_0$ to step $N-1$ is:

$$J_\pi(\mathbf{x}_0) = L_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} L_k(\mathbf{x}_k, \mathbf{u}_k) \tag{3.30}$$

where L_k is the instantaneous cost, i.e., the cost due to the application of control signal $\mathbf{u}_k$ at the step k in the state $\mathbf{x}_k$ to the dynamic system given by (3.28). The optimal cost function is the one that minimizes the total cost:

$$J^*(\mathbf{x}_0) = \min_\pi J_\pi(\mathbf{x}_0) \tag{3.31}$$

and the optimal policy $\pi^* = \{\mathbf{u}_0^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_{N-1}^*\}$ is the one that satisfy (3.31), with $J_{\pi^*}(\mathbf{x}_0) = J^*(\mathbf{x}_0)$. To exploit Bellman's principle, let's define the *tail subproblem* from step i (and state $\mathbf{x}_i$) to step $N-1$, characterized by the following cost:

$$J_\pi(\mathbf{x}_i) = L_N(\mathbf{x}_N) + \sum_{k=i}^{N-1} L_k(\mathbf{x}_k, \mathbf{u}_k) \tag{3.32}$$

and the *tail optimal policy* $\{\mathbf{u}_i^*, \mathbf{u}_{i+1}^*, \ldots, \mathbf{u}_{N-1}^*\}$, i.e., the last part of the optimal policy π^*. Bellman's optimality principle states that the policy segment from any intermediate stage to the final time step must be optimal for that subproblem. DP algorithm exploits this principle to update the optimal cost-to-go function at each node $J_k(\mathbf{x}_i)$ of the control-state grid, by proceeding *backward in time*, thus starting from the final step N up to the initial one 0. During the backward phase, it records the optimal action that minimizes the related tail subproblem:

$$\mathbf{u}_k^* = \arg\min_{\mathbf{u}_k \in \mathscr{U}_k} \left(L_k(\mathbf{x}_k, \mathbf{u}_k) + J_{k+1}\left(f_k(\mathbf{x}_k, \mathbf{u}_k)\right)\right) \; k = N-1, N-2, \ldots, 1, 0 \tag{3.33}$$

Once the backward phase is completed, $J_0(\mathbf{x}_0)$ equals the optimal cost and the sequence $\{\mathbf{u}_0^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_{N-1}^*\}$ provides the globally optimal control law. Because the computation of the cost-to-go function at every grid point must be done backward in time, DP is rarely suitable for real-time applications, as the future is always characterized by a certain degree of uncertainty. Moreover, it quickly becomes intractable as the dimensionality of $\mathscr{X}$ or $\mathscr{U}$ grows.

Pontryagin's Maximum Principle

PMP [86] provides a set of necessary conditions that any candidate policy for an optimal control must satisfy. A control input $\mathbf{u}(t)$ meeting these conditions is called extremal. Since PMP guarantees necessity but not sufficiency, every optimal control is extremal, whereas not every extremal control is optimal. Given the system dynamics in (3.24), and the cost function in Equation3.25, the corresponding Hamiltonian is defined as follows:

$$H(\mathbf{x}(t),\mathbf{u}(t),\lambda(t),t) = L(\mathbf{x}(t),\mathbf{u}(t),t) + \lambda(t)^T f(\mathbf{x}(t),\mathbf{u}(t),t) \tag{3.34}$$

here $f(\mathbf{x}(t),\mathbf{u}(t),t)$ is function that models the dynamics of the system as in (3.24), $L(\mathbf{x}(t),\mathbf{u}(t),t)$ is the instantaneous cost from (3.25), and $\lambda(t) \in \mathbb{R}^n$ is a vector of optimization variables, called co-state vector and has the same dimension of the state vector $\mathbf{x}(t)$. State and control are both bounded with constraints specified in (3.27). The minimum principle states that the optimal control law $\mathbf{u}^*(t)$ must minimized the Hamiltonian function:

$$\mathbf{u}^*(t) = \arg\min_{u\in\mathscr{U}}(H(\mathbf{u}(t),\mathbf{x}(t),\lambda(t),t)) \tag{3.35}$$

satisfying the following necessary conditions [120]:

$$\begin{cases} \dot{\mathbf{x}}^*(t) = -\left.\frac{\partial H}{\partial \lambda}\right|_{\mathbf{x}^*(t),\mathbf{u}^*(t),\lambda^*(t)} \\ \dot{\lambda}^*(t) = -\left.\frac{\partial H}{\partial x}\right|_{\mathbf{x}^*(t),\mathbf{u}^*(t),\lambda^*(t)} & \forall t \in [t_0,t_f] \\ H(\mathbf{x}^*(t),\mathbf{u}^*(t),\lambda^*(t),t) \leq H(\mathbf{x}(t),\mathbf{u}(t),\lambda(t),t),\ \forall \mathbf{u}(t) \in \mathscr{U} \end{cases} \tag{3.36}$$

along with the boundary conditions on the states that, with a fixed final time t_f, are given by:

$$\lambda^*(t_f) = \left.\frac{\partial \phi\left(\mathbf{x}(t_f),t_f\right)}{\partial x}\right|_{\mathbf{x}^*(t_f)} \tag{3.37}$$

where $\phi\left(\mathbf{x}(t_f),t_f\right)$ is the terminal cost defined in (3.25). Additionally, state constraints are enforced by specifying the admissible set of trajectories through the inequality $G(x,t) \leq 0$, such that:

$$\mathscr{X}(t) = \{\mathbf{x} \in \mathbb{R}^n | G(\mathbf{x}(t),t) \leq 0\} \tag{3.38}$$

where $G(\mathbf{x}(t),t) : \mathbb{R}^n \mapsto \mathbb{R}^p$ encodes p inequality constraints that every component of the state vector must satisfy.

PMP is a powerful tool because it converts a globally optimal control problem into local, point-wise minimization conditions. However, the global nature of the problem is hidden in the definition of the initial and boundary constraints. In addition, the necessary equations must be derived from a simplified system model, introducing errors and simplifications in the model that could lead to an optimal policy that may deviate from true optimality. Finally, PMP assumes the entire time horizon is known in advance, limiting real-time use to missions whose future evolution is either fully specified or reliably predictable.

Equivalent Consumption Minimization Strategy

Previous sections have shown that full-gorizon, global optimal control cannot be deployed in real-world applications, both for energy management and vehicle fleet coordination tasks: the computational effort is prohibitive, and uncertainties always characterize the future. A common approach to overcome these issues is to reduce the optimization windows, or even collapse it into an instantaneous optimization task, so that the problem can be solved in real-time with reduced stochasticity. For the energy management problem, a widely adopted local method is the ECMS [121, 87]), which is a static technique that instantaneously optimize the powetrain energy flow, achieving sub-optimal results, while being implementable on an ECU. The basic idea is to a multi-objective optimization problem, in which the optimization variable is an *equivalent fuel consumption*, made by the actual engine fuel consumption, $\dot{m}_f(t)$, and a virtual fuel consumption, relying on the use of the battery, $\dot{m}_{el}(t)$, which is locally minimized at each time step.

$$\dot{m}_{f,eq}(t) = \dot{m}_f(t) + \dot{m}_{el}(t) = \dot{m}_f(t) + s(t)\frac{P_{batt}(t)}{Q_{LHV}} \tag{3.39}$$

here $P_{batt}(t)$ is the power provided ($P_{batt} > 0$) or absorbed ($P_{batt} < 0$) by the battery; Q_{LHV} [MJ/kg] is the fuel lower heating value; $s(t)$ is an equivalence factor that converts battery energy into an equivalent mass of fuel. Given a vehicle and powertrain model (see Section 3.1.2 for details), solving the above single-step optimization gives the power-split command that minimizes instantaneous equivalent fuel usage. The performance of this approach, however, depended on choosing $s(t)$ correctly. In a PMP fashion, $s(t)$ plays the role of the costate associated with the battery SoC; embedding this adjoint variable in the instantaneous optimisation

enforces the terminal SoC constraint and thus establishes a formal link between ECMS and PMP [122]. Becasue the optimal $s(t)$ depends on both the powertrain architecture and the driving mission to perform, a poor estimate can compromise the performance benefits that ECMS would otherwise provide. Thus, perfect tuning of $s(t)$ theoretically requires a-priori knowledge of the entire mission and it's performed through a numerical optimization procedure. In practice, this future information is unavailable, so many adaptive schemes have been proposed to update $s(t)$ online [123]. The most common approach relies on (3.40), in which the update rule consider the past value of the equivalence factor and a proportional update to the difference of the actual SoC from the target one.

$$\begin{aligned} s_{k+1} &= \frac{1}{2}(s_k + s_{k-1}) + k_p(SoC_{trg} - SoC(t)), \\ t &= kT, \quad k = 1,2,\dots \end{aligned} \tag{3.40}$$

where s_k is the current the equivalence factor; s_{k-1} is the value used in the previous interval; s_{k+1} is the value for the next interval; SoC_{trg} is reference SoC (usually the initial for charge sustaining operation); k_p is the proportional gain of the feedback controller; T is the adaptation period. Through this adaptation rule, it is possible to achieve the final SoC target without any knowledge of future driving conditions. The only tuning that should be performed relies on the initial guess for $s(t)$ and for k_p to weight how much the SoC can deviate from the target.

Model Predictive Control

MPC [124] is an optimization-based framework that, at every sampling instant t_k, solves a *finite-horizon* optimal-control problem, placing itself midway between a global strategy and an instantaneous one. The method is particularly attractive when limited preview information is available, because over the prediction window it produces a control sequence that is optimal for the current forecast. For example, considering the energy management problem, a short-term speed predictor can usually supply a short-sighted future speed trajectory, and an MPC can then evaluate the resulting fuel consumption and SoC trajectory over that window and choose the optimal power-split [125, 126]. Likewise, in an AMoD context, demand forecasters are able to estimate customer arrivals in the next minutes, giving MPC enough

foresight to dispatch and recharge vehicles effectively [127]. Formally, after time discretizations, the controller solves:

$$\min_{\mathbf{u}_0,\dots,\mathbf{u}_{N-1}} \phi(\mathbf{x}_N) + \sum_{k=0}^{N-1} L_k(\mathbf{x}_k, \mathbf{u}_k) \tag{3.41}$$

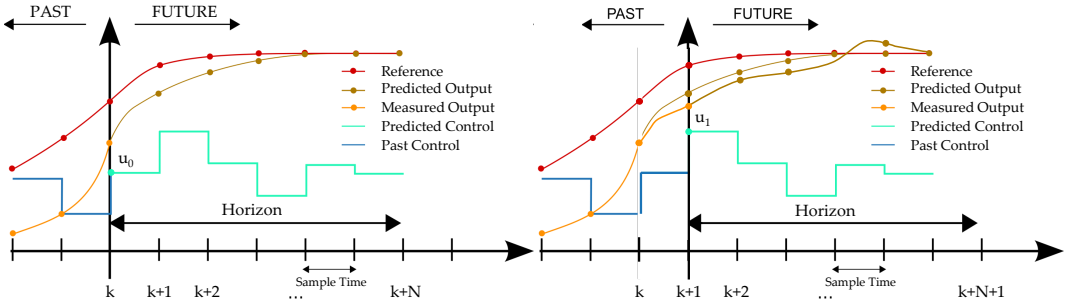
Subjected to the following constraints:

$$\begin{cases} \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad k = 0,\dots,N-1 \\ \mathbf{x}_k \in \mathscr{X}_k \quad k = 0,\dots,N-1 \\ \mathbf{u}_k \in \mathscr{U}_k \quad k = 0,\dots,N-1 \\ \psi(\mathbf{x}_N) = 0 \end{cases} \tag{3.42}$$

where Equations 3.41 and 3.42 are the time-discretized versions of Equations 3.25, 3.26, and 3.27 over the control horizon, and with $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ enforcing the system dynamics. In the simplest setting the dynamics are linearized around the current operating point, using the following approximation of the system:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k \tag{3.43}$$

so that, if the cost function is quadratic, the optimization becomes a Quadratic Program (QP) that can be solved efficiently with standard solvers. If integer decision variables are present (for instance, discrete gear selections or engine on/off modes), then the problem becomes a Mixed-Integer Quadratic Program (MIQP) or, more generally, a Mixed-Integer Linear Program (MILP) when costs and dynamics are linear. More complex variants replace the linear dynamics model with a nonlinear mapping, like a NN, at the cost of a more complex Non-Linear Program (NLP) to be solved. Because forecasts is uncertain and model simplification is affected by an intrinsic error, MPC adopts a receding-horizon implementation: at each step, the optimization is solved, but only the first input $\mathbf{u}_k$ of the optimal sequence is applied to the system, after which a new observation of the system is taken and the horizon slides forward. Figure 3.8 visualizes this loop: the left figure shows the predicted output (yellow) tracking the reference (red) over the window $k+1,\dots,k+N$, while the green line represents the optimal control trajectory: the right one the right panel repeats the procedure one sample later, starting from the newly measured state. In



(a) (b)

Fig. 3.8 RDE - (a) MPC optimization procedure over the control horizon starting from step k; (b) MPC optimization procedure over the control horizon starting at step $k+1$.

this way, MPC continuously corrects its predictions, compensates for disturbances and modeling errors, and ensures constraint satisfaction throughout the operation.

In short, MPC combines constraint handling, limited look-ahead and continual re-optimisation in a single framework, being able to achieve very robust performance to model errors and future stochasticity, making it ideal for problems where a few seconds or minutes of reliable forecast are available yet hard safety or operational limits must never be violated, like in the two control problems (i.e., HEVs EMS and robotaxi fleets coordination) that are reported in the section.

3.2.3 Heuristic-based Approaches

In practical applications, controllers are often implemented using a set of predefined rules that choose control actions based on the observed or measured states of a dynamic system. These RB approaches aim to optimize a performance index or mimic optimal control policies. Unlike approaches derived directly from optimal control theory, RB methods do not depend on explicit formal models but instead rely on heuristic rules developed through engineering intuition and experience. When applied to fleet coordination tasks, RB strategies commonly involve monitoring the current distribution of vehicles and demand and adjusting the fleet accordingly. Some approaches utilize predictive models to forecast demand [128]. Examples of typical RB fleet control strategies include equally distributed vehicles throughout a city, rules designed to anticipate future passenger requests, or strategies aimed at

balancing vehicle distribution by relocating idle vehicles to zones where departures have occurred [129]. These methods can also be combined after careful calibration of their decision logic. In the context of hybrid powertrains, RB strategies typically seek to maintain the powertrain in high-efficiency operating regions through conditional logic (such as if-then-else structures), informed by efficiency maps or fuzzy logic methods, as in the example reported in Figure 3.9.

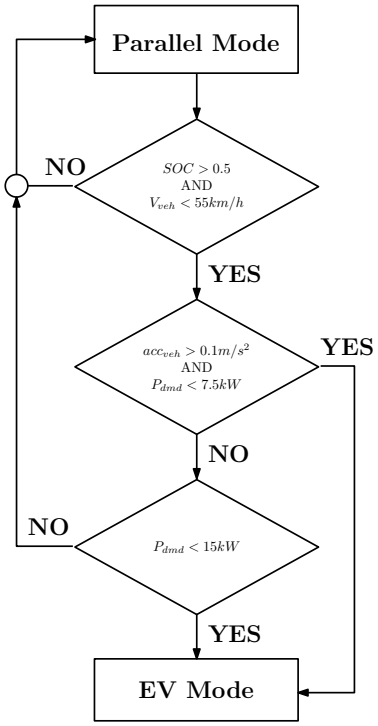


Fig. 3.9 An example of RB control.

Generally speaking, RB controllers rely exclusively on instantaneous measurements and must respect local operational constraints, including limits on parameters like power, torque, speed, vehicle numbers per region, and inter-region flows. Calibration of RB controllers usually involves parameter tuning guided by optimal control solutions. Methods such as DP may serve either as benchmarks to verify the performance of RB strategies [123] or as references for setting appropriate control rules [130, 131]. The primary advantage of RB approaches lies in their simplicity and low computational complexity. However, the presence of numerous thresholds and parameters complicates the calibration process, limiting the generalization capabilities across diverse scenarios. Consequently, RB controllers do not guarantee optimal solutions nor necessarily satisfy integral constraints.

3.2.4 Learning-based Approaches

Learning-based approaches, particularly RL combined with NNs, have emerged as valid alternatives for managing complex, stochastic, and high-dimensional systems, effectively addressing the limitations inherent in optimization-based and heuristic strategies.

As previously detailed in Section 2.2, RL frames control tasks as DDPs, in which an agent learns optimal decision-making policies through interaction with its environment, aiming to maximize the expected cumulative return. This approach naturally accommodates dynamic and uncertain environments, eliminating the necessity for explicit future knowledge required by optimization-based methods. Moreover, RL does not rely on predefined heuristic rules; instead, it implicitly captures domain expertise through extensive data-driven learning processes.

Optimization-based methods typically demand accurate predictive models and intensive computational resources, making them computationally demanding and difficult to scale for real-time applications. Conversely, heuristic approaches, despite being computationally efficient and straightforward to implement, often deliver suboptimal outcomes and limited adaptability to novel scenarios. RL methods provide a balanced alternative by combining near-optimal decision-making with computational efficiency, offering strong generalization capabilities across diverse scenarios and operational conditions.

In the context of AMoD fleet coordination, RL-driven controllers leverage extensive simulations to progressively refine strategies for vehicle repositioning, passenger dispatching, and charging management. Deep RL methodologies, such as DQL, AC architectures, and notably SAC, have shown significant potential in effectively navigating extensive state–action spaces. Advanced architectures like AC combined with GCNs leverage spatial network structures, reducing dimensionality while capturing inter-zone dependencies within city-scale networks [1, 53]. Techniques such as offline RL and meta-RL further enhance sample efficiency and policy generalization, addressing the persistent challenges associated with transferring learned policies from simulation to real-world scenarios [26, 54]. Offline RL involves training policies using previously collected datasets without additional interaction with the environment during training. By leveraging existing data, offline RL significantly reduces the risk and cost associated with real-time exploration, making it particularly

suitable for applications where live interaction with the environment is costly, dangerous, or infeasible. Meta-RL, or meta-learning [84] applied to RL, aims to enhance an agent's adaptability by training it to rapidly adjust to new tasks or environments based on previous experiences. This approach provides RL agents with the capability to generalize across diverse and previously unseen conditions, effectively improving policy robustness and transferability.

For hybrid powertrain EMS, RL approaches aim to optimize power-split strategies by implicitly learning torque distributions suitable for varying driving conditions. Initially, ML applications to EMS were limited to NNs directly predicting optimal control actions, lacking the full advantages of RL algorithms. Traditional feed-forward networks provide static representations with limited expressive capacity for dynamic state sequences. Consequently, LSTM networks emerged as a significant advancement, effectively modeling temporal dependencies within dynamic driving cycles. These LSTM-based approaches achieved high accuracy by capturing the complexity inherent in sequential data. However, supervised learning methods, including LSTM, require labeled datasets, typically generated by computationally expensive global optimization methods such as DP. Despite their performance, LSTM-based controllers trained on DP-generated data exhibit limited generalization to unseen driving conditions, prompting researchers to adopt more flexible and robust RL-based strategies.

Initially, RL methods faced challenges related to state-action space discretization. However, integrating deep learning techniques into RL significantly advanced continuous-state and action space handling. Algorithms such as DDPG, TD3, and ultimately SAC have overcome earlier limitations, notably incorporating entropy regularization to achieve robust exploration and stable training [101, 104].

A primary strength of learning-based methods lies in their effective generalization across varied operational contexts without explicit recalibration, a notable improvement over optimization-based and heuristic approaches. However, ensuring robust real-world performance requires meticulous training, sufficient exploration of state space, and integration of realistic noise and uncertainty in training environments. Although learned policies typically execute rapidly with minimal computational overhead, the training phase remains computationally intensive, often requiring specialized hardware or cloud-based computational resources.

In summary, learning-based approaches represent an integrative solution that harmonizes the optimality of model-based optimization methods with the simplicity and computational efficiency of heuristic strategies. Their demonstrated success in both AMoD fleet coordination and hybrid powertrain EMS highlights the potential for unified RL frameworks to effectively manage complex, cross-domain control challenges.

3.3 Scale 1: AMoD Systems

Within the thesis-wide framework of *RL for Mobility Across Scales*, this section addresses the *network/system* layer that complements the *in-vehicle* EMS of Scale 2, which is reported in Section 3.4. Building on the modeling and control foundations set in Sections 3.1, and 3.2, the decisions to city-wide AMoD coordination scale is analyzied, where dispatch, and rebalancing must adapt to stochastic demand and congestion while remaining computationally tractable at scale. In this cross-scale view, Scale 1 shapes trip-level operating conditions, spatiotemporal vehicle availability, travel times, and duty cycles—that Scale 2 exploits on board; conversely, vehicle-level energy usage informs system-level sustainability metrics, closing the loop between network policy and powertrain behavior.

The AMoD problem is cast as a networked optimal-control task with a high-dimensional state describing zone-level inventories, queues, and demand forecasts, and a structured action space representing rebalancing and service flows on a time-expanded graph. Consistent with Section 3.1.1, SUMO's mesoscopic engine is adopted to stress the trained policy on a realistic traffic scenario: demand is injected via OD arrivals, and congestion and travel times emerge from queue-based link dynamics that feed back into AMoD decisions. Optimization references (time-expanded flows solved with LP/MIP) connect to Section 3.2.2, while calibrated heuristics mirror Section 3.2.3. In line with Section 3.2.4, we then instantiate a SAC-based hierarchical controller that compresses the action space through graph-structured representations; a GCN encodes local topology and congestion so the actor–critic can share information across adjacent zones, stabilize training via entropy regularization, and deliver fast, rolling-horizon inference suitable for large fleets.

The agenda for this scale proceeds as follows. Section 3.3.1 formalizes AMoD coordination as a network-flow control problem and details the hierarchical SAC/GCN

policy, together with the SUMO-in-the-loop training and evaluation protocol. Section 3.3.2 instantiates the framework on a Luxembourg network (Section **??**), including demand scenarios and aggregation levels. Section 3.3.3 reports key KPIs—passenger waiting time, served demand, vehicle-kilometers traveled, and energy/emissions proxies—and then stress-tests generalization in Section 3.3.3 along three axes: *Across Space* (changes in network and aggregation), *Across Time* (time-of-day and demand shifts), and *Across Simulator Fidelity* (mesoscopic→microscopic transfers). This structure mirrors the chapter's cross-scale philosophy: principled modeling, paired optimization/learning baselines, interpretable graph-aware policies, and rigorous robustness assessments that connect network-level coordination to vehicle-level energy outcomes.

3.3.1 Methodology

This section presents a scalable AMoD coordination architecture that couples exact optimization with RL. The coordination task is first cast as a network-flow problem over the transportation graph (served-trip flows x_{ij}^t, rebalancing flows y_{ij}^t, edge costs c_{ij}^t); then a three-stage hierarchy is introduced: (i) a convex dispatch layer assigns vehicles to requests, (ii) a learning layer predicts a per-region desired allocation $\hat{\mathbf{x}}^{t+1}$, and (iii) a minimum-cost flow layer computes feasible rebalancing actions (Sections 3.3.1–3.3.1). The learning layer is implemented by a graph-based policy trained with a SAC agent, selected for stable training under stochastic dynamics and for entropy-regularized exploration that aligns with rolling-horizon control. Since operational cost c_{ij}^t typically includes a distance-proportional component, minimizing unnecessary travel directly reduces both total vehicle-kilometers traveled and fleet-wide emissions; accordingly, emissions are tracked as a primary KPI and can be incorporated in the reward as a distance-weighted penalty, ensuring that system profit and sustainability are optimized jointly while remaining compatible with SUMO-in-the-loop evaluation (Section 3.3.1).

Formulating AMoD Coordination as a Network Flow Problem

To formally describe the robo-taxi coordination task, the AMoD operator is modeled as responsible for dispatching and repositioning a fleet of N_{veh} autonomous, on-demand vehicles within a transportation network represented by a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$.

The nodes $\mathscr{V}$ denote spatially aggregated regions (e.g., pickup/dropoff zones), while the edges $\mathscr{E}$ capture admissible interconnections between regions. The number of regions is denoted by $N_{\text{reg}} = |\mathscr{V}|$, and the planning horizon is discretized into $\mathscr{T} = \{1, \dots, T\}$ intervals of length ΔT.

At each time step t, trips between origin–destination pairs $(i, j) \in \mathscr{V} \times \mathscr{V}$ are requested, with demand d_{ij}^t and associated price p_{ij}^t. Vehicle travel between two connected nodes requires $\tau_{ij}^t \in \mathbb{Z}_+$ steps and incurs cost c_{ij}^t, both depending on congestion dynamics as described in Section 3.1.1. Passengers are assumed to be impatient, leaving the system if not matched within $\tau_{\max}$ time steps. The operator must therefore orchestrate two intertwined decisions: (i) assigning available vehicles to passenger trips, and (ii) redistributing idle vehicles across regions to pre-position capacity where it will be needed.

This problem can be formulated as a network flow optimization [132], with decision variables $x_{ij}^t \in \mathbb{N}$ (number of trips served between i and j at time t) and $y_{ij}^t \in \mathbb{N}$ (number of empty rebalancing trips from i to j). The operator's objective is to maximize total profit:

$$\max_{\{x_{ij}^t, y_{ij}^t\}} \quad \sum_{t=1}^{T} \sum_{i,j \in \mathscr{V}} \left[(p_{ij}^t - c_{ij}^t) x_{ij}^t - c_{ij}^t y_{ij}^t \right], \tag{3.44a}$$

subject to demand limits, flow conservation, initial conditions, and non-negativity constraints:

$$x_{ij}^t \le d_{ij}^t, \qquad \forall i, j \in \mathscr{V}, t \in \mathscr{T}, \tag{3.44b}$$

$$\sum_{i \in \mathscr{V}} \left(x_{ij}^{t-\tau_{ij}^t} + y_{ij}^{t-\tau_{ij}^t} \right) = \sum_{k \in \mathscr{V}} (x_{jk}^t + y_{jk}^t), \qquad \forall j \in \mathscr{V}, t \in \mathscr{T}, \tag{3.44c}$$

$$x_{ii}^0 = x_i^{\text{init}}, \;\; y_{ii}^0 = 0, \qquad \forall i \in \mathscr{V}, \tag{3.44d}$$

$$x_{ij}^t, y_{ij}^t \ge 0. \tag{3.44e}$$

While conceptually well-posed, the complexity of this problem scales as $\mathscr{O}(|\mathscr{V}|^2 T)$, which severely limits its applicability in real-time large-scale scenarios. Moreover, it abstracts away stochastic elements such as variability in demand and travel times, which are crucial for real-world robustness. These limitations motivate the use of RL to complement optimization: RL can adaptively capture non-stationary patterns, while optimization ensures that constraints are enforced at each decision step.

Hierarchical Decomposition for Policy Design

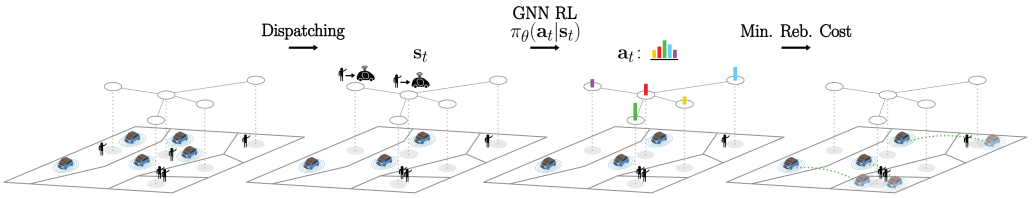


Fig. 3.10 Illustration of the proposed hierarchical decomposition for AMoD fleet coordination. Given the current distribution of idle vehicles and customer transportation requests, the decomposition entails: (1) assigning idle vehicles to trip requests (i.e., x_{ij}^t) by solving a convex dispatching problem; (2) determining a desired future allocation of vehicles across regions (i.e., $\hat{\mathbf{x}}^t$) via RL; and (3) converting $\hat{\mathbf{x}}^t$ into actionable rebalancing trips (i.e., y_{ij}^t) while minimizing the overall rebalancing cost [1].

To combine the strengths of optimization and learning, a hierarchical policy decomposition is adopted. Rather than directly mapping system states to all flow variables, the decision-making process is structured into three stages, as shown in Figure 3.10:

1. *Dispatching*: assign vehicles to passenger requests by solving an assignment problem that maximizes profit while respecting demand and vehicle availability.

2. *Desired Vehicle Distribution*: compute a compact, per-region target allocation of vehicles $\hat{\mathbf{x}}^{t+1}$ for the next time step. This intermediate representation serves as a "goal state" that guides rebalancing decisions and reduces the dimensionality of the action space from $\mathcal{O}(|\mathcal{V}|^2)$ to $\mathcal{O}(|\mathcal{V}|)$.

3. *Rebalancing*: translate the desired allocation into feasible vehicle flows by solving a minimum-cost flow problem subject to regional vehicle conservation.

This decomposition ensures scalability, since optimization handles local constraints and short-term assignments efficiently, while the RL component learns to steer the system towards desirable long-term equilibria.

Graph-Based Policy Representation

A key design choice concerns the representation of the RL policy responsible for predicting the desired vehicle distribution. Because the underlying transportation system is naturally structured as a graph, GNNs are employed to encode both local node features (e.g., demand, idle vehicles) and connectivity (e.g., congestion-weighted travel times). For additional details on GNNs, refer to Section 2.1.3. This yields three main benefits:

- *Permutation invariance*: the policy depends on node attributes rather than arbitrary indexing of regions.
- *Locality*: graph convolutions naturally propagate information between adjacent regions, enabling transfer across different network topologies.
- *Alignment with network optimization*: GNNs are structurally compatible with flow-based formulations, facilitating learning of effective coordination strategies.

Learning Signal The RL layer is trained online by interacting with the previously introduced SUMO-in-the-loop environment. Actor–critic methods, and in particular SAC [61], are a natural fit because they estimate policy gradients without differentiating through the embedded optimization subproblems that enforce short-term feasibility.

Concretely, we train a SAC agent on a reward that trades off demand satisfaction against operational effort:

$$r(t) = \sum_{i,j \in \mathscr{V}} x_{ij}^t \left(p_{ij}^t - c_{ij}^t\right) - \sum_{(i,j) \in \mathscr{E}} y_{ij}^t \, c_{ij}^t, \tag{3.45a}$$

$$c_{ij}^t = \beta \, \tau_{ij}^t, \tag{3.45b}$$

where x_{ij}^t and y_{ij}^t denote passenger and rebalancing flows (as defined earlier), p_{ij}^t is the realized fare or value of service, and c_{ij}^t is a time-varying travel cost proportional to the travel time τ_{ij}^t. The scalar $\beta > 0$ modulates the sensitivity to operational effort: larger β emphasizes frugality in empty movements, whereas

smaller β biases the policy toward higher service levels. In Section 3.3.3 we show how sweeping β systematically shifts the operating point along this cost–service frontier.

SAC then maximizes the entropy-regularized return

$$\mathbb{E}\left[\sum_t \gamma^t \left(r(t) + \alpha\, \mathscr{H}(\pi(\cdot \mid s_t))\right)\right] [61],$$

which encourages robust, high-entropy policies that adapt to stochastic demand and congestion while remaining compatible with the optimization-based dispatch and rebalancing layers.

Integration Across Scales Overall, the methodology yields a principled integration of optimization and learning: convex formulations guarantee short-horizon feasibility and safety, while RL captures the long-horizon, stochastic structure of fleet management. This mirrors the thesis-wide theme of *RL for Mobility Across Scales*: AMoD coordination (Scale 1) sets spatiotemporal operating conditions for vehicle-level energy management (Scale 2), allowing decisions at each scale to be both locally consistent and globally coherent.

3.3.2 Case Study

This case study operationalizes the framework of Section 3.3.1 in a congestion-responsive setting to evaluate the hierarchical AMoD controller under realistic conditions. A mesoscopic SUMO environment for the city of Luxembourg is adopted, using a 24-hour calibrated scenario with exogenous background traffic and signalized intersections; the original road network is spatially aggregated into regions to define the node set $\mathscr{V}$ that sustain the network-flow formulation (see Section 3.3.1). The decision pipeline of Section 3.3.1 is applied without modification: a convex dispatch layer assigns vehicles to requests, a graph-based policy predicts a per-region desired allocation, and a minimum-cost flow layer computes feasible rebalancing actions. The learning layer is implemented via a SAC-trained policy (Section 2.2.3), chosen for robustness to stochastic dynamics and entropy-regularized exploration. Performance is benchmarked against heuristic rules and an optimization-based algorithm, all within the same SUMO-in-the-loop protocol. Evaluation covers both economic

and sustainability objectives: profit and operational cost, as well as distance- and environment-oriented KPIs such as fleet emissions; because edge costs c_{ij}^t typically include a distance-proportional component, decreasing unnecessary travel tends to reduce emissions as well as it will be proven in Section 3.3.3.

Mesoscopic scenario of the city of Luxembourg The traffic environment is instantiated in SUMO [55] using its mesoscopic, queue-based traffic model (see Section 3.1.1 for details), with detailed infrastructure features (lanes, signals, and priority rules) and exogenous traffic demand. In this setting, link travel times τ_{ij}^t and edge costs c_{ij}^t evolve endogenously with congestion induced by dispatch and rebalancing actions, yielding a faithful yet tractable representation for the hierarchical controller in Section 3.3.1. The testbed uses a calibrated 24-hour scenario for Luxembourg derived from real traffic data [133].

The base network is aggregated into regions to define the node set $\mathcal{V}$ used throughout Section 3.3.1 and provide an aggregated representation of the city network for the AMoD control logic application. The higher fidelity of mesoscopic dynamics increases stochasticity and non-stationarity, motivating a graph-based policy trained with a SAC agent (Section 3.3.1) to stabilize learning and encourage exploratory, rolling-horizon behavior. Trip prices p_{ij}^t follow publicly available Luxembourg taxi fares[1], while operating costs c_{ij}^t are calibrated from publicly released ride-hailing cost structures[2]. Because both pricing and cost embed distance-linked components, reducing excess travel directly impacts the total distance driven by the fleet and its greenhouse gas emissions, which are tracked as primary KPIs alongside profit and waiting time.

Baseline policies To position the SAC-based, graph-informed controller relative to established alternatives, several baselines are evaluated within the identical three-stage pipeline of Section 3.3.1. In all cases, the dispatch layer computes passenger assignments x_{ij}^t; baselines differ only in how rebalancing targets or flows y_{ij}^t are produced, and are assessed under the same prices p_{ij}^t, costs c_{ij}^t, and KPI set. The following baselines are used, spanning from heuristics to optimization-based:

- **Heuristic rules.**

[1] See, for example, public fare calculators for Luxembourg.

[2] For instance, cost breakdowns disclosed by major platforms.

  - *No Rebalancing.* Idle vehicles are never repositioned; service relies exclusively on locally available supply.
  - *Equal distribution.* At each decision epoch, idle vehicles are driven toward a uniform per-region target to equalize availability across $\mathcal{V}$; the minimum-cost flow layer then translates this target into feasible y_{ij}^t.
  - *Plus-one* [129]. For every served trip departing a region, one idle vehicle (when available) is sent back to that region to counteract depletion; feasibility and costs are handled by the same rebalancing optimization.

- **Optimization-based reference.**
  - *MPC* [80]. A time-expanded MPC solves for $(x_{ij}^{t:t+H-1}, y_{ij}^{t:t+H-1})$ under perfect foresight of demand and network conditions (e.g., travel times, prices). In macroscopic regimes, where actions do not feed back into congestion, this constitutes an oracle-like upper bound. In the mesoscopic SUMO setting, however, intersection control and congestion feedback introduce variability, so realized travel times may deviate from forecasts; the MPC remains a stringent, though not literal, upper benchmark in this higher-fidelity regime.

3.3.3 Results

This section reports the empirical performance of the proposed SAC–GNN controller on the Luxembourg case study and compares it with standard baselines (No Reb., ED, P1) and an MPC oracle. We first analyze coordination during the afternoon peak (16:00–18:00), then study reward design via a sweep of the movement-penalty weight β, and finally test generalization across spatial aggregations, times of day, and simulator fidelities. Performance is assessed with a unified set of operator-, network-, and sustainability-level metrics (definitions in (3.46)–(3.49)): operator KPIs (profit, revenue, rebalancing cost) quantify monetization and repositioning burden; network KPIs (fleet utilization and average waiting time, citywide and by region) capture service quality and asset usage; sustainability KPIs (vehicle distance and CO_2 emissions) track externalities, with emissions computed from HBEFA factors for Euro 6d petrol passenger vehicles under the prevailing traffic state. Unless otherwise stated, the baseline SAC–GNN policy uses $\beta = 1$.

Performance metrics Model assessment integrates three complementary layers of performance, operator, network, and sustainability, so that profitability, service quality, and environmental footprint are evaluated in a unified way. Operator-side metrics quantify how well resources are monetized; network-side metrics reflect the quality of service delivered to riders and the utilization of vehicles; sustainability metrics track externalities that accompany vehicle movements. Together, these indicators expose the fundamental trade-offs between cost, service, and emissions that arise when rebalancing policies alter vehicle flows.

Operator perspective. Let x_{ij}^t denote the flow of served passengers from region i to j at time t, with price p_{ij}^t and operating cost c_{ij}^t. Let y_{ij}^t denote the flow of idle vehicles rebalanced from i to j. Over a horizon $t = 0, \dots, T$ and region set $\mathscr{V}$, operator key performance indicators (KPIs) are:

$$P = \sum_{t=0}^{T} \left(\sum_{i,j \in \mathscr{V}} x_{ij}^t \left(p_{ij}^t - c_{ij}^t\right) \; - \sum_{\substack{i,j \in \mathscr{V} \\ j \neq i}} y_{ij}^t c_{ij}^t \right), \tag{3.46a}$$

$$R = \sum_{t=0}^{T} \sum_{i,j \in \mathscr{V}} x_{ij}^t \left(p_{ij}^t - c_{ij}^t\right), \tag{3.46b}$$

$$C_{\text{reb}} = \sum_{t=0}^{T} \sum_{\substack{i,j \in \mathscr{V} \\ j \neq i}} y_{ij}^t c_{ij}^t. \tag{3.46c}$$

Profit P balances fare margins against rebalancing expenditures, revenue R isolates the contribution of served demand, and C_{reb} isolates the operational burden induced by repositioning. This decomposition clarifies whether gains in P arise from higher throughput, lower operating costs, or reduced repositioning.

Network perspective. To capture system-wide service quality and asset usage, let M_{match}^t and M_{reb}^t be, respectively, the number of vehicles matched to passengers and used for rebalancing at time t, and let M_{tot} be the total fleet size. Denote total demand by $D = \sum_{t=1}^{T} \sum_{i,j} d_{ij}^t$ and region-i demand by $D_i = \sum_{t=1}^{T} \sum_{j} d_{ij}^t$. The utilization and waiting-time metrics are:

$$UF_t = \frac{M^t_{\text{match}} + M^t_{\text{reb}}}{M_{\text{tot}}}, \qquad \overline{UF} = \frac{1}{T}\sum_{t=1}^{T} UF_t, \tag{3.47a}$$

$$t_{\text{wt,avg}} = \frac{\sum_{k=1}^{D} t_{\text{wt},k}}{D}, \tag{3.47b}$$

$$t_{\text{wt,avg},i} = \frac{\sum_{k=1}^{D_i} t_{\text{wt},k}}{D_i}. \tag{3.47c}$$

Here, $\overline{UF}$ summarizes how intensively the fleet is employed across time, while $t_{\text{wt,avg}}$ and $t_{\text{wt,avg},i}$ index customer experience overall and by region. These indicators reveal whether operator-side improvements are achieved by shifting load patterns that lengthen waits or underutilize assets.

Sustainability perspective. Since rebalancing changes where and how much vehicles travel, environmental externalities are explicitly tracked. Let ℓ_{ij} be the inter-region distance. Total distance and CO_2 emissions are defined as:

$$L_{\text{tot}} = \sum_{t=0}^{T} \sum_{i,j \in \mathcal{V}} \left(x^t_{ij} + y^t_{ij}\right) \ell_{ij}, \tag{3.48}$$

$$E_{\text{CO}_2} = \sum_{t=0}^{T} \sum_{i,j \in \mathcal{V}} \left(x^t_{ij} + y^t_{ij}\right) \varepsilon^t_{ij}. \tag{3.49}$$

The factor CO_2 emissions per vehicle movement ε^t_{ij} is computed according to Handbook Emission Factors for Road Transport (HBEFA) for a Euro 6d–compliant petrol passenger vehicle under the prevailing traffic state, yielding a realistic representation of emissions for typical Italian taxi fleets. In combination with (3.46)–(3.47), (3.48)–(3.49) make explicit how policy choices that raise profit or reduce waiting time may also increase vehicle kilometers or emissions, thereby clarifying the trade-offs that govern policy selection.

Coordination during the afternoon peak

The first experiment evaluates the system's ability to coordinate an AMoD fleet during the afternoon traffic peak (4–6 pm). Unless otherwise noted, the SAC agent is trained with a cost weighting factor $\beta=1$ as reported in (3.45a) and is used as the *baseline* for this section. As highlighted by the metrics in Section 3.3.2, the performance assessment is performed from a threefold perspective, considering operator, system efficiency, and environmental impact aspects.

Operator metrics Table 3.3 indicates that the SAC–GNN framework learns rebalancing policies that are nearly optimal even in a high-fidelity traffic environment. In particular, SAC–GNN attains a reward within 2.3% of the MPC and outperforms all heuristics. It serves the highest demand (comparable to ED and MPC) while cutting rebalancing costs by roughly 29% relative to ED (1353 vs. 1904), indicating more judicious use of idling vehicles. By contrast, MPC intentionally forgoes some low-margin trips to minimize empty travel, achieving the lowest rebalancing cost but a slightly lower served demand; the resulting policy is therefore more conservative and less dynamic than SAC–GNN, but still able to maximize the profit.

Table 3.1 System performance on the Luxembourg mesoscopic simulation during the afternoon traffic peak (04:00–06:00): operator perspective.

	Profit [$] (%Dev. MPC)	Served Demand [$]	Rebalancing Cost [$]
No Reb.	34686 (−7.8%)	34686	0
ED	36029 (−4.2%)	38298	1904
P1	35346 (−6.0%)	37157	1393
SAC-GNN	36742 (−2.3%)	38547	1353
MPC	37617 (0%)	38179	562

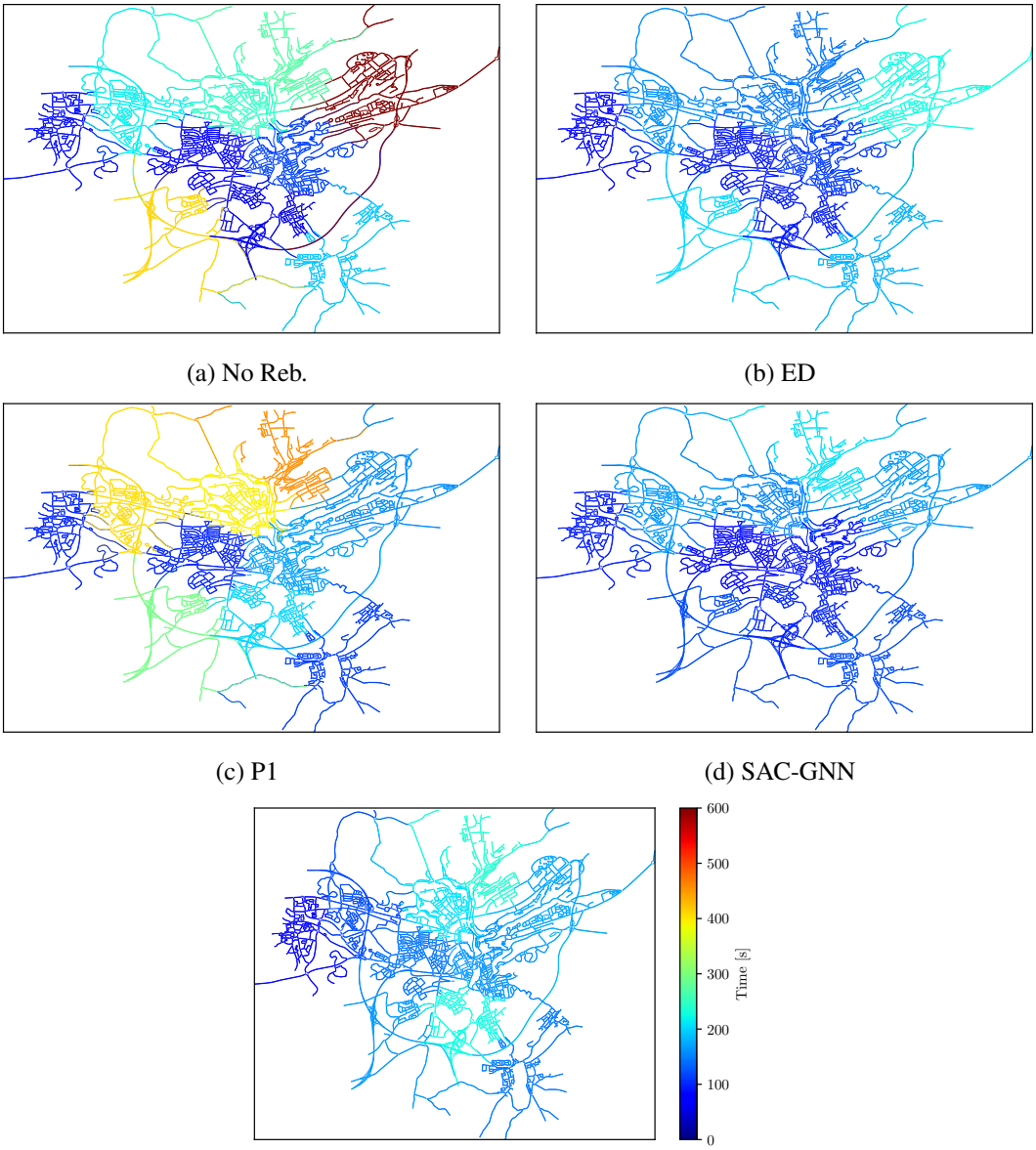
Network metrics A comprehensive assessment of network performance in Tables 3.2 and 3.3 and Figure 3.11 shows that policies differ not only in the quantity of motion they induce but in the timing and placement of that motion, which determines service quality, spatial equity, and environmental footprint. SAC-GNN delivers the lowest average waiting time (133 s) with disciplined utilization (75%), indicating

anticipatory repositioning that pre-aligns idle supply with imminent demand and contains hotspots citywide; MPC, while cost-efficient, runs the fleet at lower utilization (69%) and yields longer waits (165 s), illustrating that strict cost control can suppress responsiveness under peaky loads; ED achieves the highest utilization (80%) by aggressively relocating vehicles but pays for this in avoidable empty mileage, leading to a waiting time still 7.5% worse than SAC-GNN (143 s vs. 133 s) and demonstrating that "busier" fleets do not guarantee better service when movements are reactive rather than predictive;

Table 3.2 System performance on the Luxembourg mesoscopic simulation during the afternoon traffic peak (04:00–06:00) — network perspective.

	Average Waiting Time [s] (%Dev. MPC)	Average Fleet Utilization [%]
No Reb.	269 (+63%)	57
ED	143 (−13%)	80
P1	238 (+44%)	76
SAC-GNN	133 (−19%)	75
MPC	165 (0%)	69

Spatially, SAC-GNN, MPC, and ED all maintain broadly balanced coverage, yet only SAC-GNN couples this with superior waiting time for passengers, implying that its relocations are both geographically well-targeted and temporally well timed. In contrast, P1 and No Reb. exhibit peripheral passenger accumulation, P1 due to local, demand-chasing behavior that fails to seed low-supply outskirts, and No Reb. due to the absence of proactive balancing, yielding markedly higher waits despite utilization similar to SAC-GNN (76% and 57% respectively).



(a) No Reb.

(b) ED

(c) P1

(d) SAC-GNN

(e) MPC

Fig. 3.11 Average passenger waiting time across the city under each policy considered in the study.

Environmental impact The emissions–distance results reinforce the mechanism described in the service quality analysis for the customers: relative to MPC (0% reference), No Reb. posts the lowest emissions (-74%) and distance (16620×10^3 km) only because the fleet moves far less, at the expense of unacceptable service; among active strategies, SAC-GNN attains the most favorable trade-off, incurring a modest $+6.0\%$ increase in emissions with a proportional $+6.0\%$ increase in distance (22096 vs. 20845×10^3 km) while cutting waits by 19% relative to MPC; P1 is slightly higher on both emissions ($+6.9\%$) and distance (22304×10^3 km, around $+7.0\%$) yet delivers substantially longer waits (238 s), and ED is least eco-efficient, with the largest distance (23608×10^3 km, around $+13.3\%$) and emissions ($+13\%$) but no commensurate service gain. The near-linear scaling of emissions with distance across ED, P1, and SAC-GNN suggests comparable propulsion efficiency and points to avoidable empty vehicle kilometers as the main differentiator; SAC-GNN minimizes these by converting motion into matches rather than relocations, thereby reconciling low waiting times, contained spatial hotspots, and a moderate environmental footprint during peak-period operations.

Table 3.3 System performance on the Luxembourg mesoscopic simulation during the afternoon traffic peak (04:00–06:00) — emissions economy perspective.

	Emissions [1000 kg] (%Dev. MPC)	Distance [1000 km]
No Reb.	831 (-74%)	16620
ED	3555 ($+13\%$)	23608
P1	3360 ($+6.9\%$)	22304
SAC-GNN	3330 ($+6.0\%$)	22096
MPC	3142 (0%)	20845

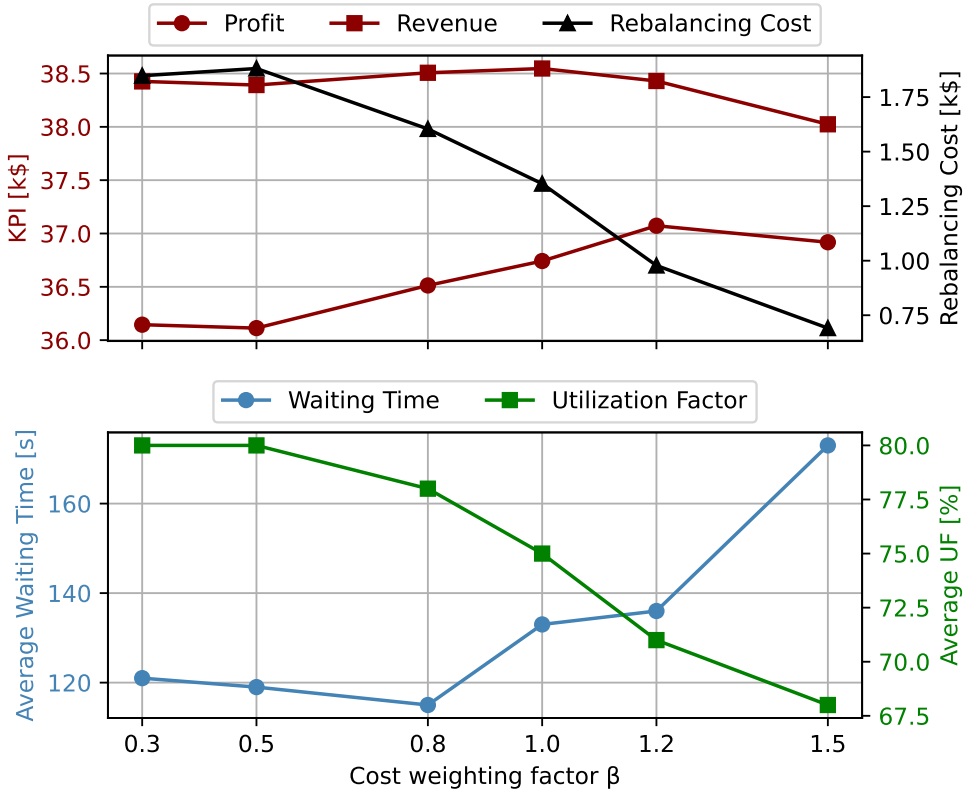


Fig. 3.12 Operator and network performance across reward structures. (Top) Profit, revenue, and rebalancing cost. (Bottom) Average waiting time and fleet utilization factor.

Reward design

Reward shaping steers the learned controller by penalizing vehicle motion through a travel-time–proportional cost, modeled on each edge (i, j) as $c_{ij} = \beta\, \tau_{ij}$ and included in the per-step reward of (3.45a). Varying the scalar weight β tilts behavior along a continuum from dynamic, service-seeking strategies (small β) to conservative, cost-averse strategies (large β), with the resulting operator- and network-level outcomes summarized in Figure 3.12.

Mechanistically, increasing β raises the opportunity cost of moving without immediate revenue, reducing the attractiveness of speculative repositioning and long deadheading; the controller therefore favors holding vehicles idle unless the expected value of a future match outweighs the penalization. This mechanism produces three practical regimes. Under light penalization ($\beta \leq 0.5$), aggressive pre-positioning keeps vehicles close to demand, sustaining high fleet utilization

and low, stable waiting times, albeit with noticeable empty-kilometer accumulation. In a moderate range ($\beta \approx 0.8$–1.0), avoidable empty trips are trimmed without materially affecting realized revenue; rebalancing cost falls sharply and operator profit peaks near $\beta \approx 1.0$, while service quality remains competitive with only mild increases in waiting time. With heavier penalization ($\beta \geq 1.2$), the fleet becomes increasingly static: anticipatory rebalancing is discouraged, spatial mismatches persist, utilization declines, and waiting times rise; in the extreme, the policy may forgo profitable matches that require nontrivial deadheading. From the operator perspective (Figure 3.12, top), rebalancing cost decreases monotonically with β, confirming that the penalty effectively suppresses empty movements; revenue is comparatively inelastic for small-to-moderate β, so profit follows a shallow concave profile, improving as wasteful movement is removed, then flattening or slightly receding once higher penalties begin to displace revenue-generating matches that require pre-movement. From the network perspective (Figure 3.12, bottom), average waiting time increases and fleet utilization decreases as β grows, reflecting the mirror image of those cost savings: fewer empty trips imply fewer anticipatory moves, so vehicles are less often in the right place at the right time; this degradation is modest at first but accelerates beyond a threshold around $\beta \geq 1.0$ in our setting. Collectively, these responses trace a Pareto frontier between operator efficiency (low rebalancing cost, high profit) and customer experience (low waiting time, high utilization): small increases in β deliver outsized cost reductions with minimal service impact, whereas beyond the knee of the curve, the marginal savings are offset by steeper increases in waiting time and erosion of utilization. The practical implication is that β serves as a single, interpretable lever to align the policy with operational priorities and can be re-tuned as conditions change; within the experiments presented, a moderate choice ($\beta = 1.0$) secures most efficiency gains while preserving competitive network performance, with $\beta < 1$ favored under strict latency targets and $\beta > 1$ acceptable when cost constraints dominate and service-level agreements are looser.

Generalization Performance

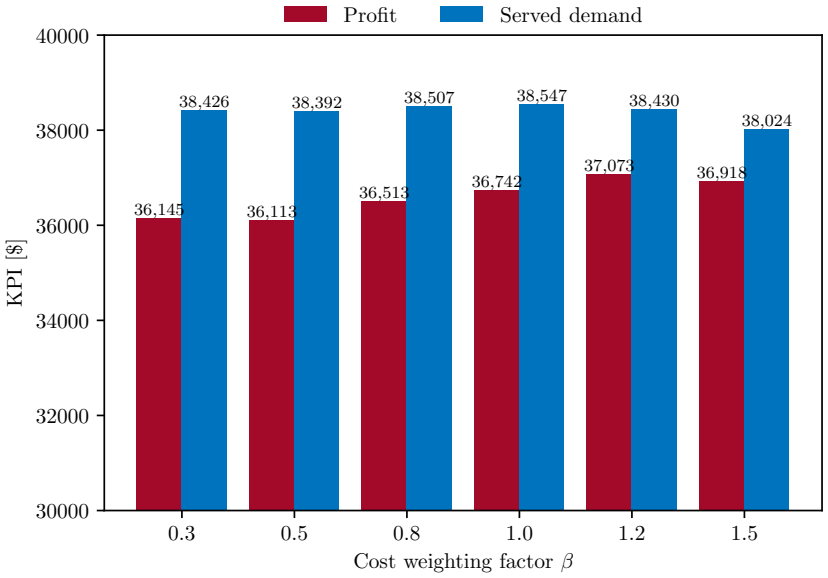
This section focuses on operator-side performance profit, served demand, and rebalancing cost since network- and environment-level responses mirror the patterns established in the afternoon-peak analysis, as detailed in Section 3.3.3, and lead to the same qualitative conclusions. Generalization is assessed for a SAC-GNN

policy trained during the afternoon peak (04 pm–06 pm) on a city aggregated into 10 regions by evaluating zero-shot transfer (i.e., without additional training) along three axes: (i) *spatial aggregation* (alternative regional partitions), (ii) *time of day* (nighttime and morning rush in addition to the afternoon peak), and (iii) *simulator fidelity* (training in the macroscopic setting and zero-shooting in the mesoscopic one). For each axis, the analysis compares the zero-shot policy (SAC-0Shot) against a retrained SAC policy when applicable, standard baselines (No Reb., ED, P1), and an MPC, reporting operator KPIs to quantify portability and robustness across changing spatial resolutions, demand regimes, and modeling fidelities.

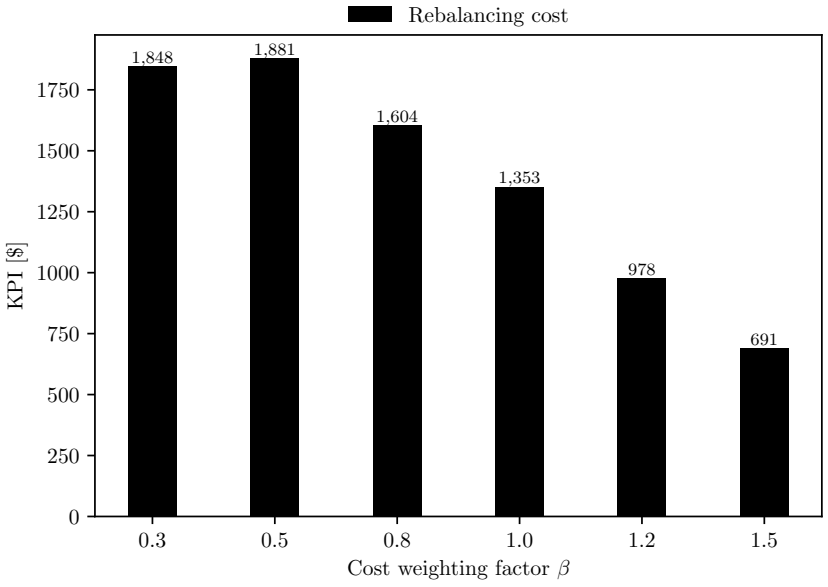
Across spatial aggregation Figure 3.13 reports operator KPIs under varying spatial aggregations, contrasting the retrained SAC policy, the zero-shot baseline (SAC-0Shot), and the MPC. The retrained SAC maintains consistent performance as the aggregation changes: profit deviates from the MPC by only 0.6% to 3.3% from smaller to larger action spaces, with a characteristic pattern of slightly higher served demand and higher rebalancing cost relative to MPC. The zero-shot baseline exhibits robust portability: despite being trained on a different aggregation, its profit gap to MPC remains moderate (3.1%–4.4%), and it preserves the same qualitative trend, more served demand than MPC, but at the price of elevated rebalancing. The excess rebalancing cost is the primary contributor to the residual profit gap versus the retrained SAC. Overall, these results indicate that the SAC-GNN architecture generalizes effectively across spatial partitions; while retraining recovers near-optimal performance, the zero-shot policy remains competitive in the presence of aggregation-induced changes to travel patterns and region-level dynamics.

Across time of day Transfer across demand regimes is assessed by applying the afternoon-peak policy to two additional periods:

- night time: 12 am–02 am, characterized by low demand, smaller fleets, and near free-flow traffic;
- morning rush: 07 am–09 am, with high demand, larger fleets, and moderate congestion, having traffic features similar to the afternoon peak.



(a) Profit and served demand



(b) Rebalancing cost

Fig. 3.13 Comparison of KPIs for the retrained SAC, its zero-shot counterpart (SAC-0Shot), and the MPC oracle under a different spatial aggregation of the network.

Table 3.4 Nighttime (12 am–02 am) operator KPIs. In addition to standard baselines, SAC-NA (afternoon+night) and SAC-NMA (night+morning+afternoon) are evaluated.

	Profit [$] (%Dev. MPC)	Served Demand [$]	Rebalancing Cost [$]
No Reb.	5297 (-19%)	5322	0
ED	5784 (-12%)	6858	1037
P1	6349 (-3.2%)	6777	393
SAC-0Shot	5411 (-18%)	6112	672
SAC-NA	6333 (-3.5%)	6752	387
SAC-NMA	6178 (-5.8%)	6509	260
MPC	6561 (0%)	6891	329

Table 3.5 Morning peak (07 am–09 am) operator KPIs. In addition to standard baselines, SAC-NA (afternoon+night) and SAC-NMA (night+morning+afternoon) are evaluated.

	Profit [$] (%Dev. MPC)	Served Demand [$]	Rebalancing Cost [$]
No Reb.	45630 (-15%)	45917	0
ED	51050 (-5.3%)	55717	4384
P1	49980 (-7.2%)	52478	2158
SAC-0Shot	51229 (-4.9%)	55789	4268
SAC-NA	49909 (-7.3%)	55725	5510
SAC-NMA	52252 (-3.0%)	55835	3306
MPC	53867 (0%)	54889	943

To probe the value of broader training corpora, SAC-NA (trained on afternoon+night) and SAC-NMA (trained on night+morning+afternoon) are also considered. Operator KPIs are summarized in Tables 3.4 and 3.5.

Zero-shot transfer is heterogeneous across regimes. At night, profit is down by up to 18% versus MPC for SAC-0Shot, reflecting a combination of lower served demand and higher rebalancing costs. This aligns with the markedly different spatial distribution of night time demand, which is more concentrated, reducing the payoff of the highly active repositioning learned on peak periods. In contrast, during the morning rush, the zero-shot policy generalizes well, with a profit gap of only -4.9% relative to MPC and an operating style aligned with the afternoon peak, characterized

Table 3.6 Cross-fidelity generalization on the Luxembourg mesoscopic simulation during the afternoon peak (04 pm–06 pm): zero-shot SAC trained in the macroscopic setting vs. baselines and a SAC trained directly in mesoscopic.

	Profit [$] (%Dev. MPC)	Served Demand [$]	Rebalancing Cost [$]
No Reb.	34686 (-7.8%)	34686	0
ED	36029 (-4.2%)	38298	1904
P1	35346 (-6.0%)	37157	1393
SAC-0Shot	36406 (-3.2%)	38472	1670
SAC (meso)	36742 (-2.3%)	38547	1353
MPC	37617 (0%)	38179	562

by a high served demand and substantial rebalancing. Incorporating night time data during training largely closes the nocturnal gap: SAC-NA cuts the profit deviation to -3.5% via a 2.1% increase in served demand and a 42% reduction in rebalancing cost relative to SAC-0Shot, though performance during the morning becomes slightly overactive, reducing profit there. Training across all three periods (SAC-NMA) yields the best overall robustness, with deviations of -5.8% (night) and -3.0% (morning), indicating that broader temporal coverage effectively regularizes behavior across regimes.

Across simulator fidelity: from macroscopic to mesoscopic Cross-fidelity transfer is examined by deploying a policy trained in a macroscopic simulator directly in the higher-fidelity mesoscopic environment (zero-shot). This pathway is attractive because macroscopic training requires approximately 17 hours versus roughly 5 days for mesoscopic training under identical settings. Table 3.6 reports the afternoon-peak results.

Cross-fidelity transfer is strong: the macroscopic-trained SAC-0Shot achieves profit within 1.1% of the SAC retrained in mesoscopic, with nearly identical served demand and only a modestly higher rebalancing cost. Moreover, SAC-0Shot outperforms all heuristic baselines in the mesoscopic setting. These findings suggest that macroscopic training captures the salient decision structure required for mesoscopic operation, enabling substantial savings in computational budget and wall-clock time without sacrificing operator performance.

To conclude and summarize the generalization performance of the proposed policy, across spatial partitions, demand regimes, and simulator fidelities, the SAC-GNN framework exhibits reliable transfer on operator KPIs. Retraining recovers near-optimal performance when needed; otherwise, zero-shot deployment is a competitive default, particularly in regimes resembling the source domain (e.g., morning vs. afternoon peaks) or when leveraging macroscopic pretraining. Expanding the training corpus across regimes further stabilizes behavior, reducing rebalancing overshoot in low-demand settings while preserving high-demand responsiveness. Given that network-level and environmental metrics track rebalancing intensity as established in Section 3.3.3, the operator-side generalization reported here is indicative of broader system-level robustness suitable for real-world deployment.

3.4 Scale 2: Energy Management of HEVs

Within the thesis-wide framework of *RL for Mobility Across Scales*, this section addresses the *in-vehicle* scale. Whereas the previous part dealt with the *system scale*, i.e., AMoD fleet control and rebalancing, the focus here shifts to the single vehicle, where a supervisory EMS splits traction demand between the ICE and the electric path while regulating battery SoC. Conceptually, the AMoD layer can be used to generate trip-level decisions (e.g., assignments, routes, horizon/ETA, expected duty cycle) and operating contexts (traffic class, stop density); Scale 2 translates those high-level, network-informed conditions into power-split actions and battery usage that minimize tailpipe CO_2 or any other performance index under real-vehicle constraints.

We ground the study on a state-of-the-art P2 pHEV available in the European market, detailing the architecture and component ratings most relevant to control design, and we assembled a comprehensive driving-cycle dataset that includes type-approval traces (NEDC, WLTC), on-road RDE measurements, and synthetic RDE-compliant profiles. The dataset is partitioned into train/test splits and spans a wide range of power demand and aggressiveness, enabling learning and out-of-distribution assessment consistent with the cross-scale agenda.

Two complementary virtual test rigs are employed: (i) a *backward kinematic* model for efficient training/sweeps and optimal reference generation, and (ii) a high-fidelity *forward dynamic* GT–Simulink co-simulation for closed-loop validation

under a very realistic simulation platform. Performance is reported primarily in terms of CO_2 [g/km], electric energy [kWh], and terminal SoC compliance.

Methodologically, we pair *optimization-based* references with *learning-based* controllers under a common protocol. DP provides cycle-optimal upper bounds with full preview; ECMS offers a real-time implementable baseline after equivalence factor calibration. On the learning side, we develop a DP-informed LSTM, trained under a supervised learning fashion, and a SAC controller with RL training method, benchmarked also against a DDQL variant. Controllers are instantiated for the two canonical regimes: *Charge Sustaining* (CS), minimizing fuel while regulating SoC around a 15% target, and *Charge Depleting* (CD), minimizing fuel while tracking a distance-parameterized linear SoC trajectory to a prescribed terminal value. In line with the cross-scale perspective, CD explicitly exploits trip-progress information that could be supplied by the AMoD layer (e.g., remaining distance/horizon), while CS emphasizes local regulation.

Beyond aggregate metrics, we "open the black box" with post-hoc interpretability maps over speed–acceleration–SoC–distance grids and run ablations to probe the role of trip progress and temporal memory (recurrent parametrizations). Finally, we stress-test generalization by comparing SAC against DP on a matrix of **10** heterogeneous cycles and **4** initial SoC levels $(90\%, 70\%, 50\%, 15\%)$, and we confirm closed-loop behavior on the forward rig. This integrated, cross-scale protocol quantifies not only optimality gaps but also policy structure, robustness, and practicality, positioning the learned EMS as the vehicle-level executor of network-level AMoD decisions.

3.4.1 Case Study

To assess and compare the performance of different EMSs, this section introduces both the vehicle on which the different control strategies have been applied and the driving cycles used for the machine learning models training and performance assessment. First, Section 3.4.1 outlines the architecture and key specifications of the reference vehicle, which is a pHEV manufactured by Mercedes-Benz, highlighting the features most relevant to control design. Then, Section 3.4.1 describes the driving cycle database used to train and validate the machine-learning models, detailing how type-approval traces, on-road measurements, and synthetic RDE profiles were combined to cover a broad spectrum of real-world operating conditions. Together,

these two subsections define the technical context in which the learning algorithms are developed and benchmarked.

Vehicle Specifications

The case study examined in this Section is the Mercedes-Benz E300de, a state-of-the-art diesel pHEV marketed in Europe. Figure 3.14 shows its drivetrain layout, while the principal vehicle and powertrain parameters are listed in Table 3.7. The car adopts a P2 architecture: a Euro-6d-Temp compliant, 2.0-liter turbocharged four-cylinder diesel engine is connectable, via an auxiliary clutch (K0), to a 90 kW permanent-magnet synchronous EM. Through a torque converter (TC) and a nine-speed automatic transmission (AT), both machines transmit torque to the rear axle. Powered by a water-cooled 13.5 kWh Li-ion battery (Li-NMC, 365 V, 37 Ah), the EM alone can propel the vehicle for up to 54 km and achieve 130 km/h in electric-only operation. A DC/DC converter steps the high-voltage bus down to 12 V, supplying the auxiliary battery and low-voltage loads.

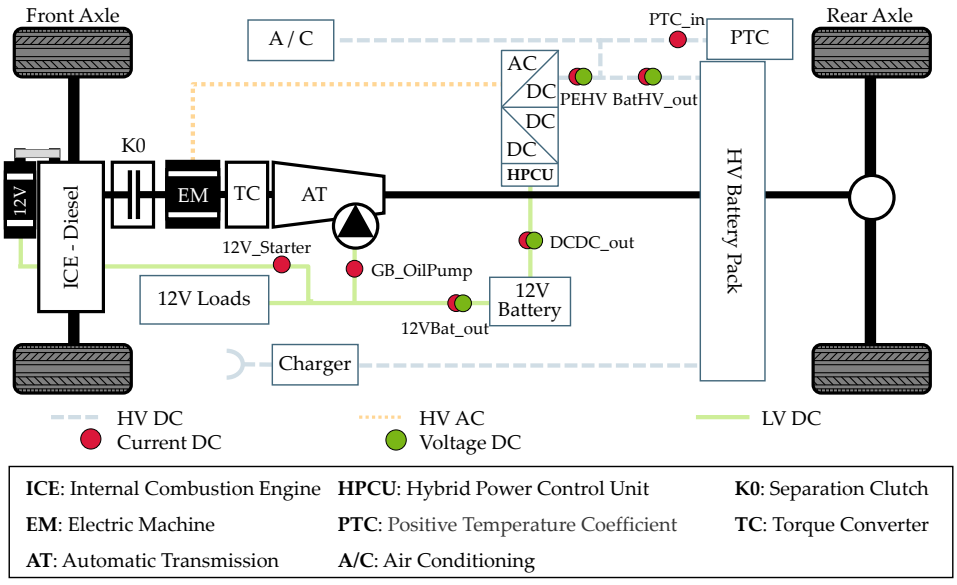


Fig. 3.14 Powertrain layout: a diesel engine is connected through an auxiliary clutch to an EM. Both the ICE and the EM are connected to the transmission by means of a torque converter.

Typically, a pHEV operates in two battery-management regimes:

Table 3.7 Vehicle and powertrain main specifications.

Vehicle	Curb Weight [kg]	2060
	Power [kW] @ 100 km/h	14.9
Transmission	Type	9-AT w/ Torque Converter
ICE	Type	In-line 4 cylinders Turbo Diesel
	Displacement [cm^3]	1950
	Max Power [kW] @ 3800 rpm	143
	Max Torque [Nm] @ 1600-2800 rpm	400
	Compression Ratio	15.5:1
EM	Type	PM Synchronous Motor
	Max Power [kW] @ 2000 rpm	90
	Max Torque [Nm] @ 1750 rpm	440
	Max Speed [rpm]	6000
HV Battery	Type	Li-NMC
	Rated Voltage [V]	365
	Cacapity [kWh]/[Ah]	13.5/37
	Cooling System	Water Cooled

- Charge Depleting (CD): The battery is sufficiently charged to propel the vehicle primarily in full electric.
- Charge Sustaining (CS): Once the SoC falls to its lower boundary, the ICE is used to propel the vehicle and keeps the SoC within a narrow band.

From a powertrain-control perspective, three additional modes can be distinguished irrespective of the SoC strategy:

- Electric drive: the EM delivers all traction torque.
- Load-point shifting: the ICE drives the vehicle while the EM is used to charge the battery, allowing the ICE to operate at a higher, more efficient load point.
- Hybrid drive: the ICE and EM jointly supply the required wheel torque.

Driving Cycles Dataset

The performance of a machine-learning-based EMS is deeply affected not only by the size of the training set but also by how faithfully the data reproduces real driving

Table 3.8 RDE-compliant driving cycle characteristics

	Urban	**Rural**	**Highway**
Percentage on total distance	29–44 %	23–43 %	23–43 %
Minimum distance [km]	16	16	16
Speed [km/h]	$v \leq 60$	$60 < v \leq 90$	$v > 90$
Average speed [km/h]	$15 < v \leq 40$	$60 < v \leq 90$	$v > 90$

behaviour [134]. For that reason we built a database that merges type-approval traces (i.e., NEDC and WLTC) with a large pool of RDE driving cycles collected on public roads during the vehicle test campaign. To enlarge the available dataset even further, we created synthetic RDE-compliant cycles following a methodology inspired by [135]. Each measured trace was split into sub-cycles that begin and end at a standstill; the resulting segments were labelled as urban, rural or highway on the basis of their speed profile. Then, randomly selected sequences of labelled fragments were stitched together until the assembled trajectory satisfied all regulatory constraints listed in Table 3.8. A small amount of white noise was then superimposed on the resulting speed signal to reproduce the natural variability that affect this phenomenon.

The final dataset comprises 80 distinct traces that span a remarkably wide range of energy demand and driving aggressiveness. Figure 3.15 illustrates this coverage by plotting every cycle against two energetic indicators: the squared mean speed on the x-axis, which scales with average traction power, and the mean speed–acceleration product on the y-axis, which reflects how aggressive a profile is. The grey area in the background, derived from 47000 km of field data on a comparable plug-in hybrid over two years [136], confirms the strong overlap between our dataset and real-world driving operation. We then divided the cycles into a training subset (grey points) and a test subset (green points), used for training and testing the ML models. A Particular emphasis is given to a RDE-compliant route [137] (red square), used in the following Sections as a test case to assess the performance of the developed methodologies.

The selected RDE trace lasts about 92 min and covers 96 km. Figure 3.16(b) shows its speed–time profile, with colours indicating the urban, rural and motorway segments, while Figure 3.16(a) maps the corresponding route reconstructed from PEMS data. As required by the regulation, an urban section is followed by rural driving and finally by motorway operation. Altogether, the diversity and realism of

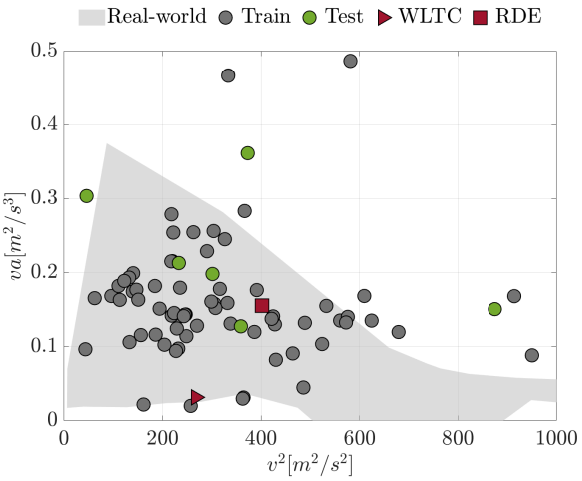
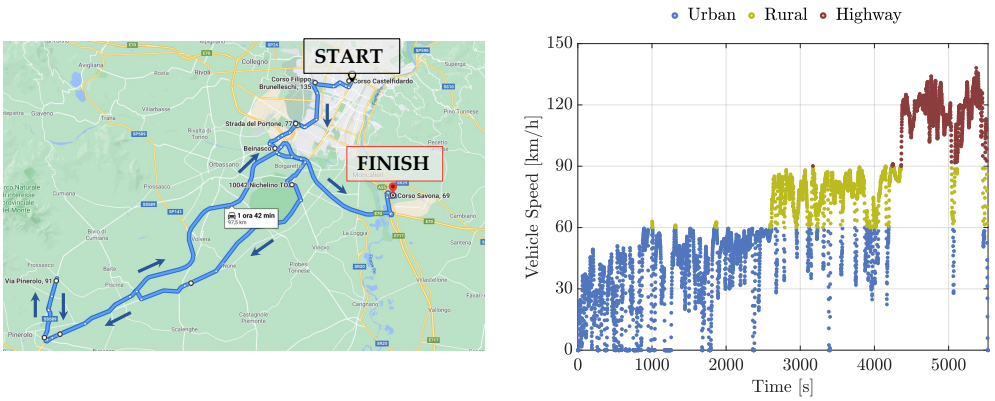


Fig. 3.15 Driving cycles dataset plotted as a function of squared vehicle speed and the product of vehicle speed and acceleration.



(a) (b)

Fig. 3.16 RDE - (a) Route obtained from PEMS, overlaid on a topographic map (Courtesy of Google Maps); (b) Vehicle speed as a function of time, divided into urban, rural, and highway segments.

the resulting dataset provide a solid foundation for training the ML models and for assessing its robustness under real-world conditions.

Virtual Test Rig

To assess the potential of proposed learning based-EMS, two complementary virtual test rigs were employed and kept consistent with the vehicle specification reported in Table 3.7. Both platforms share the same parameterization (mass, transmission, ICE/EM ratings) and rely on steady-state performance maps for the ICE, the EM, and a first-order ECM for the battery, derived from dedicated testing and data processing [138]. Additional modeling details are provided in Section 3.1.2.

Backward kinematic model. For RL training and DP benchmark evaluation, we developed a simplified digital twin based on a backward kinematic formulation. The vehicle speed trace is imposed at the wheels and the corresponding power flows are propagated upstream through the driveline to the power sources. Thanks to its limited dynamics, this approach is widely adopted for full-cycle simulations because it is computationally efficient yet provides reliable fuel-consumption estimates. These properties make it particularly suitable for RL frameworks, ensuring tractable training times while preserving physical consistency across the cycle. The virtual platform was developed in MATLAB/Simulink ®.

Forward dynamic model. To evaluate closed-loop behavior with higher fidelity, we also employed a forward, causal model of the same vehicle developed in co-simulation between GT-SUITE and Simulink ®, and validated against experimental measurements [138, 139]. In this platform, a virtual driver, modeled with a PID controller, tracks the target speed profile by issuing accelerator and brake commands. The plant solves the longitudinal and powertrain dynamics (including the P2 layout with auxiliary clutch, torque converter, and 9-speed automatic transmission), producing the actual vehicle response. Fuel consumption and electrical quantities are computed from the steady-state maps consistent with Table 3.7. This forward formulation captures transient effects (e.g., engine start/stop penalties, converter behavior, shifting latency) that are relevant for controller assessment under realistic conditions.

Use within the study. The backward model is used primarily for RL policy training and large-scale sweeps over the driving cycle dataset (Section 3.4.1), while the forward model is used to validate the trained policies and to benchmark real-time

implementable logics under closed-loop conditions. In what follows, DP is applied on the backward model to obtain reference optimal solutions, whereas ECMS and learning-based controllers are assessed on both models to balance optimality insight and realism.

3.4.2 Methodology

This section details the modeling and control framework used to develop and benchmark energy management strategies for the reference pHEV. Two complementary strands are considered under a common experimental protocol: (i) optimization-based methods that deliver cycle-optimal or near-optimal references, and (ii) learning-based methods that produce implementable, data-driven controllers. All approaches share the same vehicle parameterization (Table 3.7), the same driving-cycle dataset (Section 3.4.1), and a unified validation environment to ensure a fair, apples-to-apples comparison. Controllers are designed and assessed under both CS) and CD logics. For CS, the objective is to minimize fuel (or CO_2) while regulating SoC around a target band; for CD, the objective is to ensure a desired final SoC target while minimizing fuel. These objectives are encoded as costs/constraints for DP and ECMS, and as reward terms for RL; they also shape the supervision targets used by the LSTM approach. The remainder of the section presents the optimization-based references (DP and ECMS), followed by the learning-based controllers (LSTM and RL).

Optimization-based Framework

This subsection presents two complementary optimization-based references used as benchmarks in the study. DP provides an upper bound on achievable performance under full drive-cycle preview and thus serves as an optimality benchmark. ECMS offers a causal, real-time implementable controller that, however, requires the offline calibration of the equivalence factor to ensure the final SoC constraint. Both approaches are instantiated on the vehicle described in Table 3.7; DP is applied to the backward kinematic model to generate cycle-optimal solutions, while ECMS is deployed on both the backward and the forward dynamic virtual test rigs to assess implementability under realistic closed-loop dynamics.

Dynamic Programming. DP is a canonical tool to compute the optimal power split in HEVs. In its standard setting, the policy specifies how traction demand is apportioned between the internal combustion ICE and the EM at each discrete time step to minimize a chosen cost—typically the total energy consumption over the driving cycle [140]. The framework is flexible and can accommodate augmented objectives, e.g., including tailpipe pollutant emissions [141] or the CO_2 associated with grid recharging [142]. Despite its optimality guarantees (up to discretization), DP is unsuitable for on-board real-time use: it requires full preview of the cycle because the value function is computed by backward recursion, and the computational burden grows rapidly with grid density and problem dimensionality. Consequently, DP is adopted here as a *benchmark* to bound the performance of implementable strategies [140, 143–145].

In this work, DP is implemented following Bellman's principle of optimality [69] using the open-source MATLAB code from ETH Zürich [146] on the backward kinematic version of the virtual test rig. The problem is posed with battery SoC as the state variable and with ICE state and EM power as control inputs; SoC dynamics and actuator limits are enforced through the underlying maps and constraints consistently with Table 3.7. This configuration provides a high-quality reference solution against which the proposed EMS strategies are evaluated.

ECMS. ECMS computes real-time control actions by minimizing an instantaneous equivalent fuel cost that balances fuel use and electrical energy exchange through an equivalence factor (see (3.39)). In this study, the ICE power is selected as the control variable, and the instantaneous cost is formulated by combining the fuel term with the electrically equivalent term scaled by the equivalence factor, while enforcing SoC and actuator constraints through the same maps used for DP. To obtain an optimal ECMS setting, the equivalence factor is calibrated via a genetic algorithm [147], targeting the desired SoC objective (CS/CD) over the driving-cycle set.

ECMS is deployed on both the backward kinematic model and the forward dynamic virtual test rig. The backward implementation enables efficient sweep and calibration, while the forward implementation assesses closed-loop feasibility under realistic dynamics. This dual deployment ensures that ECMS serves as a strong, implementable optimization baseline against which learning-based controllers are compared in Sections 3.4.3 and 3.4.4.

Learning-based Framework

This section presents the two learning-based EMS approaches developed in this work and evaluated under a common experimental protocol: (i) a *supervised* LSTM-based controller that mimics DP solutions on a broad set of driving cycles, and (ii) an *RL* controller that learns a policy directly from a task-oriented reward under CS or CD objectives. Both approaches use the dataset defined in Section 3.4.1 and are validated in the same forward dynamic virtual test rig to ensure comparability. The LSTM path provides a DP-informed, implementable baseline; the RL path explores policy search without supervision, enabling flexible objective shaping via the reward.

LSTM-based EMS. This work adopts a supervised-learning strategy in which LSTM networks [148] learn offline to approximate near-optimal power-split policies from DP targets computed over a broad set of driving cycles (see Section 3.4.1). This methodology is comprehensively described in [38]. LSTMs are chosen for their ability to capture long-range temporal dependencies while mitigating the vanishing/exploding gradient issues typical of conventional RNNs. In online operation, the trained models deliver sub-optimal yet implementable policies with a markedly lower computational burden than DP.

The methodological pipeline, summarized in Figure 3.17, comprises: (i) DP-based target generation on a backward kinematic vehicle model; (ii) offline supervised training of LSTM networks using DP state–action pairs; and (iii) online validation by integrating the trained controller into the forward dynamic virtual test rig. To reflect distinct battery-management objectives, separate models are trained for CS and CD operation.

As shown in Figure 3.18, a two-network structure is employed in both regimes: a *classification* network predicts the ICE ON/OFF state, while a *regression* network outputs the ICE torque. This decomposition improves torque fidelity whenever the engine is active. For the CS case, inputs include variables directly related to engine-state prediction and torque demand (e.g., vehicle speed/acceleration, power demand, engine speed/state). For the CD case, mission-progress indicators (e.g., distance/time to destination or their normalized fractions) are additionally provided so that the policy aligns with near-linear, distance-based SoC depletion when feasible. The classification head returns the ICE state; the regression head returns the ICE torque command.

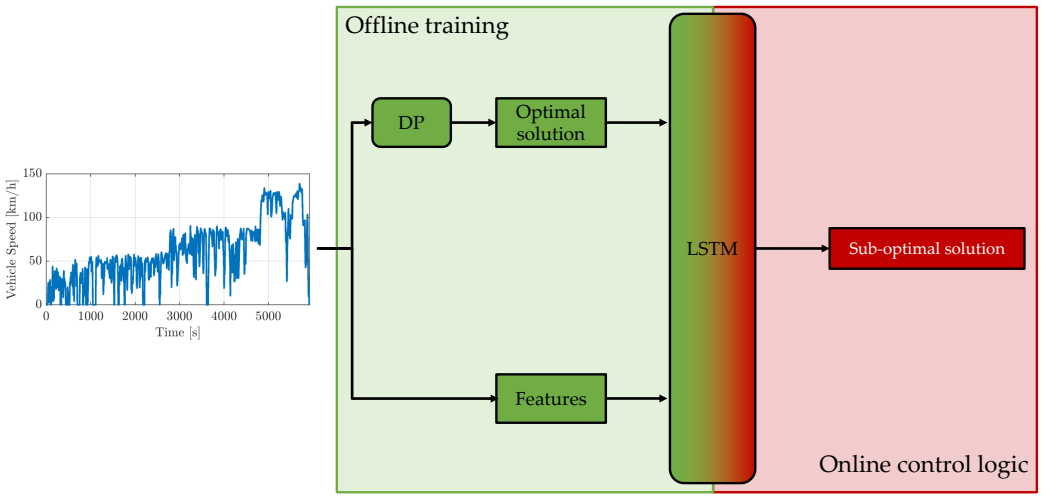


Fig. 3.17 LSTM-based EMS workflow. DP generates optimal state–action pairs for offline supervision; the trained LSTM policy is then evaluated online within the forward dynamic virtual test rig.

The training dataset combines type-approval and RDE cycles and is augmented with synthetic RDE-compliant traces to broaden coverage; mutually exclusive train/validation/test splits are used to mitigate overfitting. Network hyperparameters (depth, number of units, regularization, learning rate, dropout, etc.) are selected via *Bayesian optimization* [149] using the MATLAB Deep Learning Toolbox. The resulting LSTM controller serves as a DP-informed, implementable baseline for the comparative analysis in Section 3.4.3 and Section 3.4.4.

RL-based EMS. The RL methodology follows the general framework of Section 2.2. Unless otherwise stated, the configuration reported here refers to the *baseline agent* used in the results. As illustrated in Figure 3.19, the state vector is

$$\mathbf{s} = \left[v_{\text{veh}},\ a_{\text{veh}},\ \omega_{\text{gb,in}},\ SoC,\ d_{\text{veh}}/d_{\text{veh,tot}}\right],$$

i.e., vehicle speed, vehicle acceleration, gearbox input speed, battery SoC, and the fraction of distance traveled over the total cycle. This selection provides the agent with both instantaneous operating conditions and a measure of cycle progress, enabling phase-aware decisions. Alternative state definitions are compared in Sections 3.4.3 and 3.4.4 to assess their impact on performance. The action is the normalized

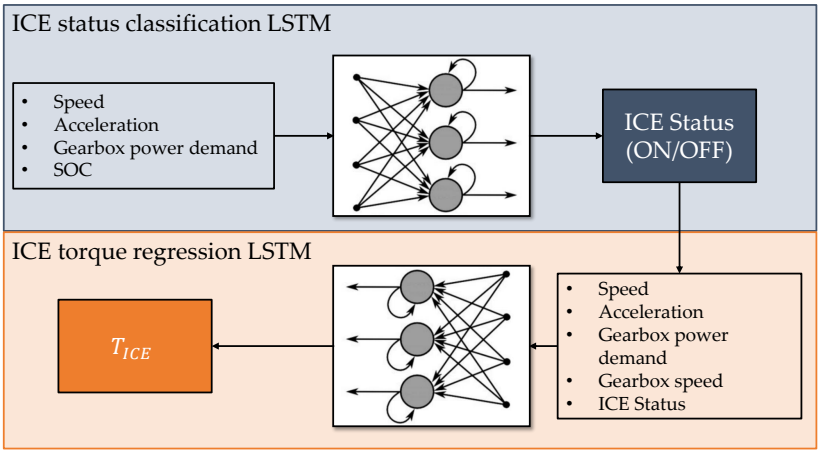


Fig. 3.18 Two-network EMS architecture: ICE state classification (ON/OFF) and ICE torque regression, adopted for both CS and CD regimes.

engine torque request,

$$a = \frac{T_{\text{ice}}}{T_{\text{ice,max}}} \in [0,1],$$

which prevents unfeasible torque commands by construction and typically improves training stability and convergence.

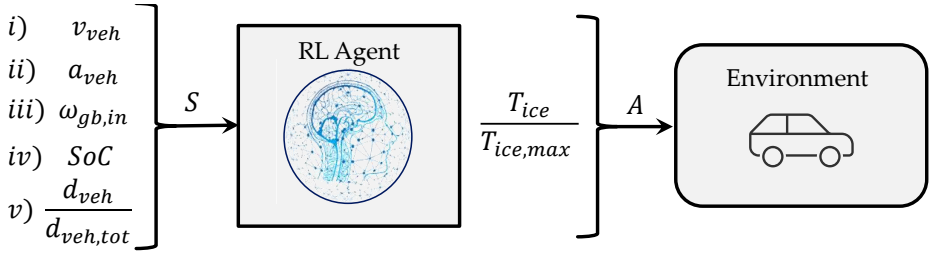


Fig. 3.19 RL environment for the EMS problem. The baseline agent interacts with a MATLAB®/Simulink® simulation, receiving the state vector and returning the normalized engine-torque action.

The reward balances fuel minimization with battery charge objectives under either CS or CD operation, using soft penalties on SoC because RL cannot impose hard terminal constraints. For *CS*, a fixed SoC target $SoC_{\text{trg,CS}}$ is enforced via a quadratic penalty:

$$r = k_1 - k_2 \dot{m}_f \Delta t - k_3 \left(SoC - SoC_{\text{trg,CS}}\right)^2, \tag{3.50}$$

where $\dot{m}_f$ is the instantaneous fuel mass flow, and $k_1, k_2, k_3 > 0$ are weights. The fuel term drives consumption reduction; the quadratic term regulates SoC about the target. The offset k_1 can keep the reward non-negative.

For *CD*, the reference follows a linear depletion profile with respect to traveled distance. Let d_{veh} be the distance at time t, and $d_{\text{veh,tot}}$ the total cycle distance. With initial SoC SoC_0 and desired end-of-cycle value SoC_{end} (set to 15% in this work), the reference trajectory is

$$SoC_{\text{trg,CD}} = SoC_0 - \left(SoC_0 - SoC_{\text{end}}\right) \frac{d_{\text{veh}}}{d_{\text{veh,tot}}} \tag{3.51}$$

The per-step reward is then

$$r(t) = \begin{cases} k_1 - k_2 \dot{m}_f \Delta t - k_3 \left(SoC - SoC_{\text{trg,CD}}\right)^2, & \text{if } SoC_{\min} < SoC < SoC_{\max} \\ -k_4, & \text{if } SoC \geq SoC_{\max} \\ -k_5, & \text{if } SoC \leq SoC_{\min} \end{cases} \tag{3.52}$$

with $k_1, k_2, k_3, k_4, k_5 > 0$. Using distance rather than time in (3.51) decouples the reference from local speed fluctuations, yielding a consistent depletion trajectory across heterogeneous cycles. As in CS, k_2 and k_3 are tuned to trade-off fuel saving against adherence to the SoC schedule; k_4 and k_5 discourage violations of the admissible SoC range.

Training uses the driving-cycle dataset of Section 3.4.1. To promote robustness (especially under CD), the initial SoC is randomly selected between 50% and 95% at the beginning of each episode. Episodes are early-terminated upon crossing $SoC_{\min}$ or $SoC_{\max}$. We train for 2000 episodes and select hyperparameters via an extensive grid search. For comparative studies, the same reward structure (CS or CD) and protocol are applied to all RL agents to ensure fairness. For completeness, parameter values for the DDQL baseline agent are initialized from [95] and subsequently refined for the SAC agent. For a more detailed explanation of the RL-based freamwrok please refer to [95, 2, 106].

The LSTM-based controller leverages DP supervision to provide a strong, implementable baseline. The RL-based controller, sharing the same plant interface,

dataset, and evaluation protocol, optimizes task-driven rewards that explicitly encode CS or CD goals. This paired design enables a controlled comparison in the results sections (Sections 3.4.3 and 3.4.4) between DP-informed supervision and direct policy search under identical conditions.

3.4.3 Charge Sustaining Policy

In a pHEV, the EMS must accommodate a two-stage battery-depletion paradigm considering both CD and CS operation. This section focuses on the CS logic, developed using the RL methodology of Section 3.4.2 and the driving cycle dataset of Section 4.2, which is partitioned into training and test sets to assess out-of-distribution behaviour and generalization. The presentation proceeds from algorithmic benchmarking to mechanism-level insight and, finally, validation: we first benchmark SAC against DDQL under a common setup to motivate SAC as the baseline controller; we then characterize the baseline SAC trained for CS with a 15% SoC target, analyzing reward evolution and the joint trajectories of SoC and fuel consumption to establish convergence and stability; next, we open the "black box" via an interpretability study over a structured grid of speed, acceleration, SoC, and trip progress, reporting both the control action (normalized ICE torque) and the discrete operating mode (EV, E-boost, load-point moving); building on these findings, we probe the role of trip progress by ablating the distance input and retraining, and we investigate temporal memory by parameterizing the policy and value function with an LSTM; finally, we validate the learned controllers on a high-fidelity forward virtual test rig, evaluating CS performance, constraint compliance, and fuel/CO_2 outcomes and contextualizing results against DP and ECMS references.

SAC vs DDQL

To evaluate the training performance in a CS scenario, the cumulative reward trends of the SAC and DDQL agents were compared. Both agents were trained with the same negative reward function and an initial SoC of 50% to ensure a fair comparison. As shown in Figure 3.20, SAC (red) clearly outperforms DDQL (blue) in terms of sample efficiency, reaching convergence after approximately 140 episodes, whereas DDQL requires around 1700 episodes to achieve comparable stability. This significant difference in convergence speed can be attributed to SAC's explicit policy modeling, which contrasts with the value-based action selection of DDQL. Additionally, SAC leverages clipped double Q-learning and adds noise to the target actions, mitigating overestimation and overfitting issues, and thus enhancing learning capability.

In the initial training phase (about the first 100 episodes), DDQL exhibits a smoother and more gradual increase in cumulative reward, while SAC shows wider

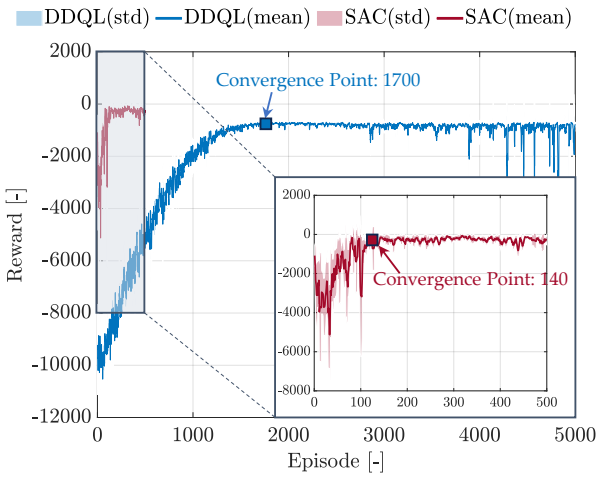


Fig. 3.20 Training curves of SAC (red) and DDQL (blue) agents in a CS scenario, plotted as cumulative reward versus training episodes. [2].

fluctuations. These fluctuations are linked to the deliberate exploration encouraged by action noise and the stochasticity introduced by the multi-cycle training dataset. Nevertheless, this variability diminishes as SAC approaches convergence, delivering superior stability and performance.

To assess not only sample efficiency but also the achievable performance, the two agents were evaluated on the WLTC driving cycle, which is part of the training dataset. In this comparison, two additional benchmark strategies were included: DP, used as the optimal reference, and the ECMS, representing a local optimization approach commonly adopted in real-time applications with its equivalent factor adaptive version (see Section 3.2.2 for more details). As illustrated in Figure 3.21, SAC consistently outperforms DDQL in terms of the trade-off between CO_2 emissions and final SoC. SAC achieves lower emissions for a given SoC target, approaching the performance of DP while being very close to the performance of the ECMS, while not requiring any a-priori parameter tuning, like it was done with the ECMS policy reported in the plot. These results confirm that SAC is not only more sample efficient but also delivers higher quality control policies. Consequently, SAC was selected as the baseline agent for the subsequent analysis, and retraining was performed to match the desired SoC target after the charge-depleting phase.

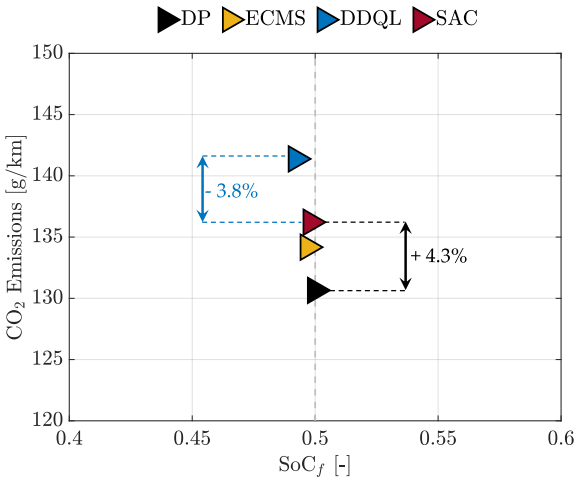


Fig. 3.21 Performance comparison on the WLTC cycle between DP (black), ECMS (yellow), DDQL (blue), and SAC (red) in terms of CO_2 emissions versus final SoC. [2].

Consequently, SAC was selected as the baseline agent for the subsequent analysis, and retraining was performed to match the desired SoC target after the CD phase, which is set to 15%.

Baseline Policy Training Assessment

Building on the results presented in the previous section, a new SAC agent was trained to optimize fuel consumption in a CS framework. In this configuration, the objective is to maintain the battery SoC at a target value of 15%, starting from a depleted condition. The training process is evaluated by analyzing both the evolution of the episode mean reward and the trends of fuel consumption and SoC across episodes.

The cumulative reward evolution, reported in Figure 3.22, serves as a compact indicator of the agent's learning progress. In the initial training phase, the mean reward exhibits a steep increase, reflecting a rapid adaptation of the control policy despite the significant variability observed across episodes. After approximately 400 episodes, the curve stabilizes around a plateau, while the shaded area representing the standard deviation becomes relatively narrow. This behaviour indicates that the agent has converged towards a stable and robust policy, with no signs of late-stage divergence.

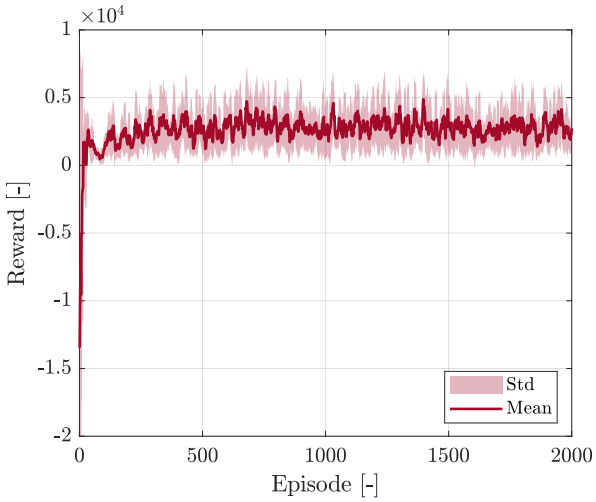


Fig. 3.22 Training reward as a function of episodes. The solid line denotes the mean value across 20 episodes, while the shaded area represents the corresponding standard deviation.

Additional insights into the learning process are provided in Figure 3.23, which shows the episode trajectories of battery SoC (top) and cumulative fuel mass m_f (bottom), color-coded from early episodes (dark red) to later ones (bright yellow). In the early stages of training, the SoC trajectories display high dispersion and may exhibit large deviations from the target, occasionally exceeding 0.30 or approaching the lower operational bound—an indication of an immature and unrefined strategy. As training progresses, the trajectories become increasingly consistent and converge towards the desired terminal value of 15%, confirming the agent's ability to meet the terminal SoC constraint while avoiding boundary violations and premature episode terminations. Similarly, the fuel consumption profiles evolve from irregular and scattered patterns in early episodes to smoother, more predictable trends with reduced variability in later stages. The parallel improvement in SoC regulation and fuel consumption behaviour demonstrates that the learned policy effectively coordinates battery utilization and engine operation in a CS strategy, ensuring compliance with the SoC constraint over the entire driving cycle.

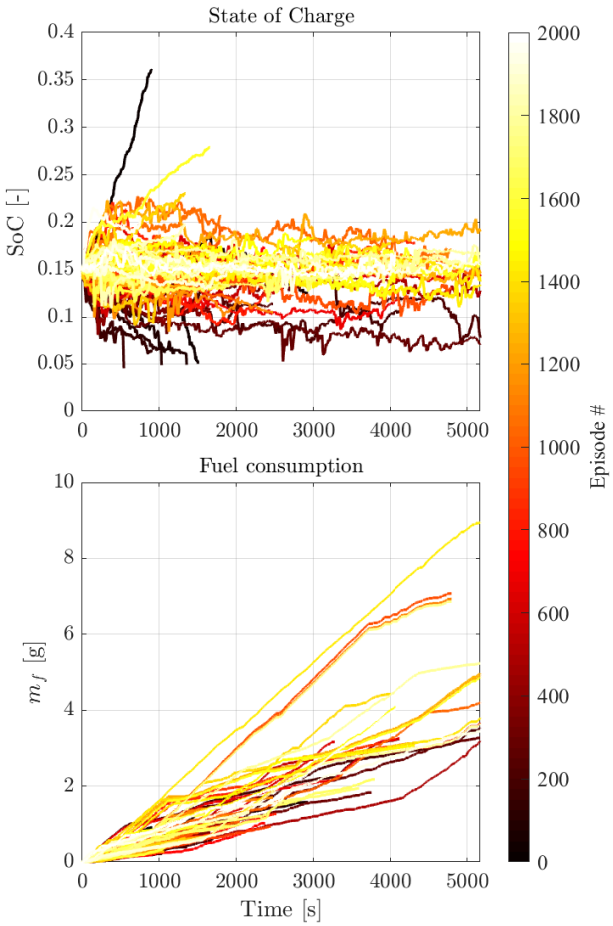


Fig. 3.23 Evolution of battery SoC (top) and cumulative fuel consumption (bottom) during training episodes. The colour scale represents the episode index, from dark red (early episodes) to bright yellow (late episodes).

Decision-Making Process

Understanding how an RL policy makes decisions is essential when pHEV energy management. Unlike RB or optimization-based strategies, RL controllers are often treated as black boxes, providing limited visibility into why a given action is chosen in a specific operating condition. To address this limitation, a post-hoc interpretability analysis is reported, which is designed to expose the internal decision logic of the trained agent. Specifically, the learned policy was investigated over a strcured grid of vehicle speed, acceleration, battery SoC, and trip progress, as the remaining distance, and record both the control action (normalized ICE torque request), reported in Figure 3.30 and the discrete operating mode (EV, E-boost, load point moving), shown in Figure 3.25. The SoC-distance grid used is reported in Table 3.9.

Table 3.9 Combinations of battery SoC and normalized travelled distance used for the policy-interpretability analysis under CS operation.

Normalized traveled distance	SoC level
20%	10%
	15%
	20%
50%	10%
	15%
	20%
80%	10%
	15%
	20%

Concerning the ICE exploitation strategy, the maps are strongly structured by SoC, while the trip progress seems to have a very marginal effect in the CS logic. At low SoC (10%), the commanded engine fraction is high over almost the entire speed-acceleration plane, indicating an aggressive use of the ICE to sustain/recover charge. Only a narrow band at the lowest accelerations calls for limited engine torque when the torque requested by the driver is low. For SoC values close to the target (15%), the policy becomes selective. For low–moderate accelerations, the engine request is small, while a clear activation threshold appears as the acceleration increases; above that threshold, the engine contribution ramps up with both acceleration and speed, in order to satisfy the increased torque demand. At a higher value of SoC (20%),

the engine demand is essentially zero for all the tested vehicle-acceleration points, highlighting a preference for pure-EV operation when the battery buffer is available.

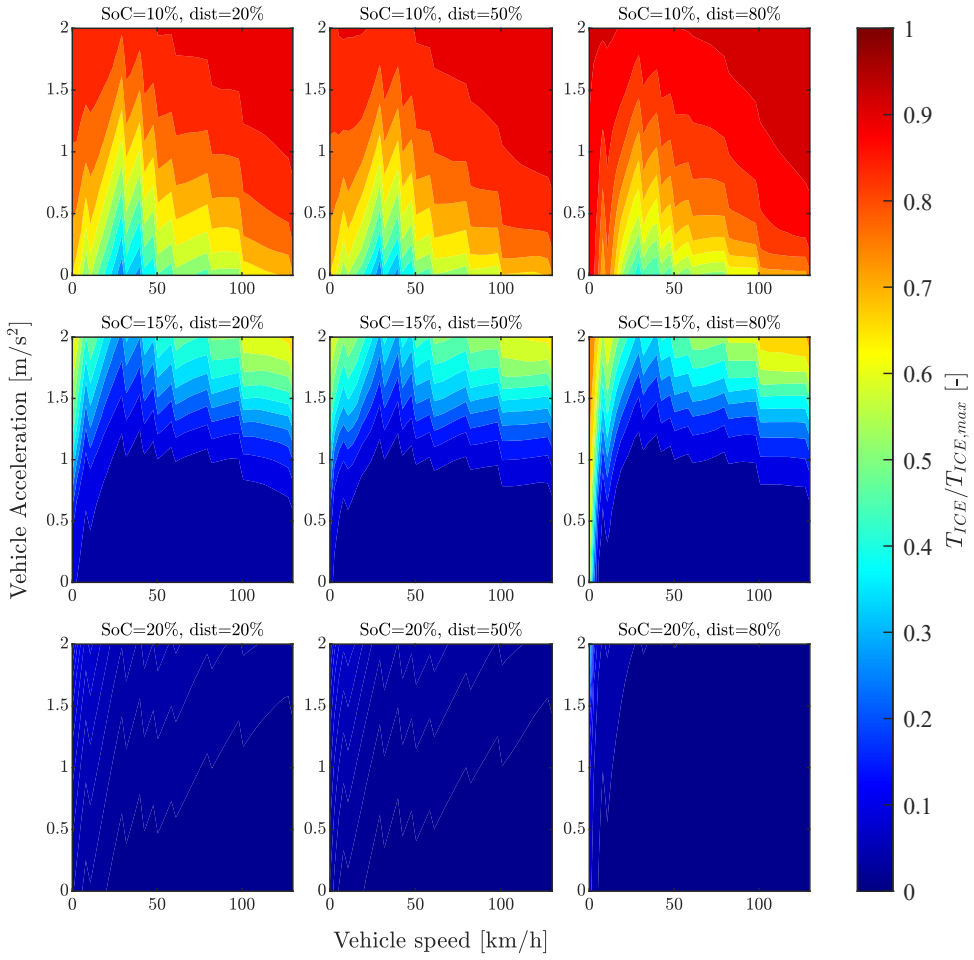


Fig. 3.24 Grid of normalized action distributions for nine combinations of SoC and travelled distance percentage; each panel shows the normalized engine torque within its corresponding SoC-distance bin for the CS logic.

The operating modes map confirms the ICE torque trend. At low SoC (10%), E-boost prevails at medium–high torque request to satisfy performance, and load-point moving emerges at lower accelerations and medium speeds, where the engine can be operated at higher loads to sustain/recharge the battery in a more efficient way. EV operation is largely suppressed. At close-to-target SoC value (15%), EV

dominates for low–mid accelerations, while E-boost appears at higher accelerations, where transient power demand exceeds what the electric path alone would supply efficiently. Load-point moving is marginal. Finally, when there is available electric energy (SoC = 20%), the policy selects EV mode across the board, independently of speed and acceleration. Also in this case, the effect of the trip progress is very marginal.

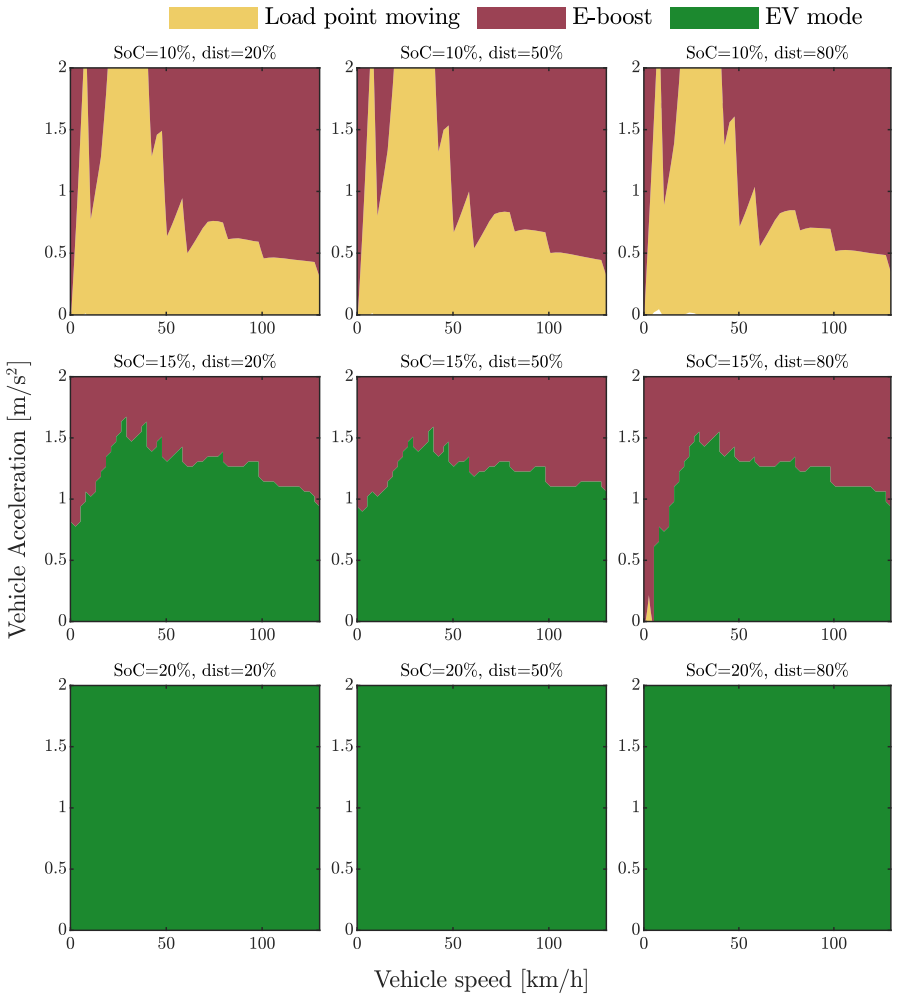


Fig. 3.25 Grid of speed–acceleration plots for nine combinations of SoC and travelled distance percentage; each panel shows the engine management strategies within its respective SoC and distance interval for the CS logic.

Overall, the two figures show a coherent CS strategy: the agent prioritizes EV operation whenever the SoC margin allows it, engages the ICE primarily under high torque demand, and uses load-point moving at low demand to maintain SoC when the battery is depleted and operate the ICE in more efficient operating points. The smooth, monotonic shifts of the mode boundaries with SoC indicate a well-structured policy rather than episodic behaviors.

Effect of Vehicle Distance

Motivated by the limited influence of trip progress observed in the interpretability maps of Figures 3.30–3.25, we conducted an ablation study in which the vehicle distance was removed from the state vector. The resulting agent (named SAC-wo-distance hereafter) was trained and compared against the baseline SAC. Figure 3.26 shows nearly indistinguishable training dynamics: both agents exhibit comparable convergence rates and steady-state episode rewards, with similar variability. This suggests that excluding distance does not affect learning stability or asymptotic performance.

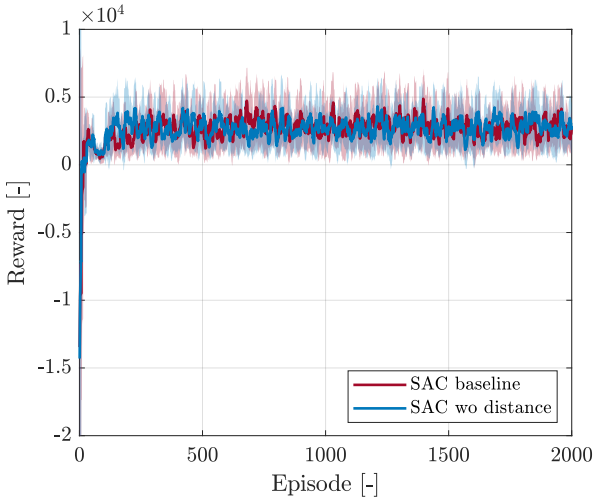


Fig. 3.26 Training reward evolution for the SAC controller: comparison between the baseline agent (red) and the variant without the distance input (blue). Solid lines denote the episode-mean reward, while the shaded bands indicate the corresponding standard deviation.

To assess closed-loop behaviour, both policies were deployed on a backward kinematic test rig following an RDE-compliant cycle and benchmarked against DP. As reported in Figure 3.27, both SAC controllers regulate SoC close to the CS target of 15%. Both policies maintain the SoC close to the target throughout the entire cycle. The baseline policy exhibits a slightly deeper initial battery discharge followed by a recovery towards the end, whereas the ablated policy adopts a more conservative profile. However, the resulting differences in the SoC trajectories are marginal. The DP profile shows the expected globally planned pattern with deeper mid-cycle depletion and a deliberate recovery near the end. In terms of efficiency, all strategies

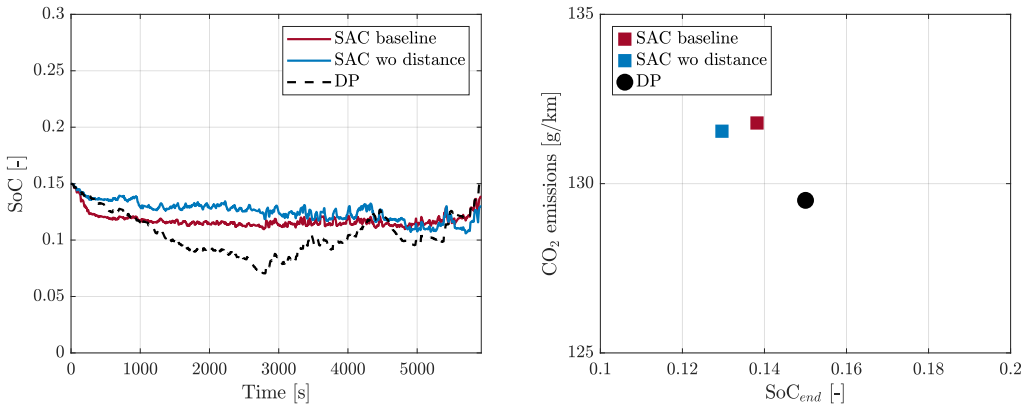


Fig. 3.27 RDE driving–cycle performance comparison of the SAC baseline (red), the SAC without the distance input (blue), and a DP benchmark (black). In (a), the SoC trajectories over time are reported, while in (b) the trade-off between the CO_2 emissions and the terminal SoC.

terminate close to the target SoC; DP achieves the lowest fuel/CO_2 emissions, while the two SAC variants are clustered nearby with only a small penalty.

Overall, within the CS operating condition and for the tested cycle, the ablation results indicate that trip progress (distance) is largely redundant in the state representation: removing it leaves training behaviour, SoC regulation, and emissions essentially unchanged. This finding confirms the interpretability analysis, which revealed only a marginal role of distance in the policy's decision making process.

Effect of Recurrence

DP is used as a benchmark because it solves the fully observable MDP: it has access to the entire driving profile (past and future) and can therefore compute a globally optimal energy management policy. By contrast, the baseline RL formulation, implemented with fully connected networks, operates under partial observability: the state only encodes the current driving condition, with no explicit memory of the past and no preview of the future. This effectively turns the problem into a Partially Observable Markov Decision Process (POMDP). To increase effective observability, one idea could be the introduction of temporal memory by parameterizing both the policy and the value function with an LSTM and retraining the agent (hereafter, SAC

recurrent). This agent was then compared with the baseline in terms of training behaviour and closed-loop performance on a backward kinematic test rig, following the same evaluation protocol used previously. As shown in Figure 3.28, the two agents exhibit nearly identical learning dynamics: similar convergence rates, comparable steady-state episode rewards, and overlapping variability bands. The recurrent parameterization, therefore, does not materially affect training stability or asymptotic reward.

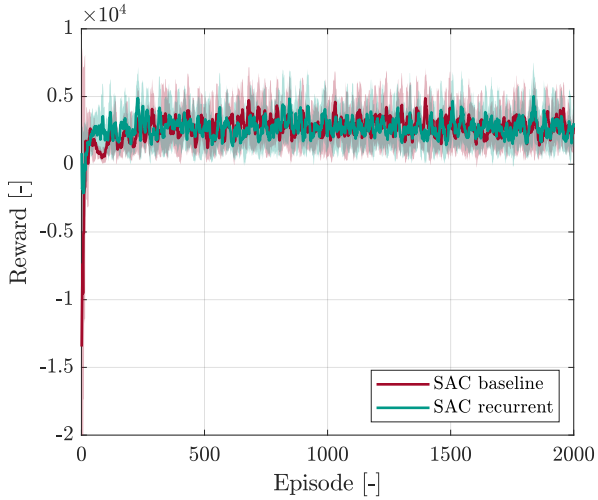


Fig. 3.28 Training reward evolution for the SAC baseline agent (red) and the SAC recurrent variant (teal). Solid lines represent the episode-mean reward, while shaded areas indicate the corresponding standard deviation.

On the RDE-compliant cycle (Figure 3.29), the SAC recurrent agent regulates SoC closer to the CS target (15%) for most of the drive and then intentionally depletes near the end, finishing slightly below the target. This greater exploitation of the battery buffer yields a -1.5% reduction in tailpipe CO_2 relative to the baseline SAC. The baseline, instead, allows a larger mid-cycle deviation from the target and compensates late, closer to DP behaviour.

To understand this different behaviour, in Figure 3.30 the decision making process in terms of control action distribution is reported for the recurrent policy. The first eye-catching consideration is that, with recurrent information, the trip progress plays a key role, while in the baseline policy, this state information was useless. In the 15% SoC row, the recurrent agent commands substantially higher ICE torque over a broader speed-acceleration region than the baseline, whose maps are largely in

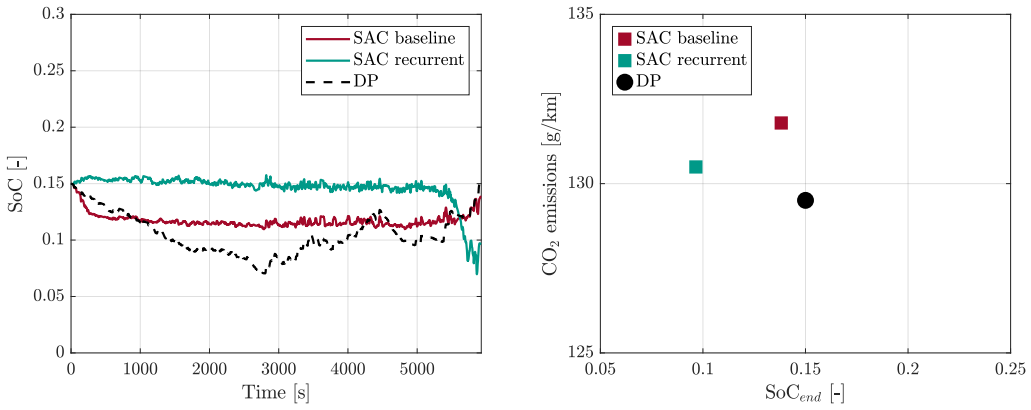


Fig. 3.29 RDE driving–cycle performance comparison of the SAC baseline (red), the SAC recurrent variant (teal), and a DP benchmark (black). In (a), the SoC trajectories over time are reported, while in (b), the trade-off between tailpipe CO_2 emissions and terminal SoC.

the low-torque/EV zone. Likewise, at 20% SoC the recurrent policy still requests non-zero torque in many operating points, while the baseline remains almost entirely EV. This systematic, moderate ICE usage when SoC is at or above the target prevents mid-cycle battery depletion and keeps the state close to 15%. Conversely, the baseline's strong EV preference at 15–20% SoC allows SoC to drift below the target, necessitating a late recovery. Near the end of the trip (right column), the recurrent maps show a reduction of ICE demand at 15–20% SoC, which enables the agent to spend the remaining battery energy to discharge the battery in the final steps of the cycle; the baseline maps instead expand the ICE-on region, consistent with charging back if the SoC is below the target. Overall, the recurrent architecture yields smoother, more continuous decision boundaries and a proactive regulation strategy: hold SoC near the setpoint during the drive, then exploit the residual buffer at the end, while the baseline exhibits a more reactive pattern that tolerates mid-cycle deviation and compensates late.

Despite the modest CO_2 benefit and the more structured, temporally consistent decisions revealed by the maps, the aggregate performance of the two SAC policies remains very similar. Moreover, the recurrent agent exhibits a weaker enforcement of the terminal SoC constraint (ending below 15%), which, together with the added implementation and tuning complexity of LSTMs, limits the practical value of the

recurrent parametrization for CS operation in this case study. Consequently, the baseline SAC offers a favorable trade-off between simplicity and performance.

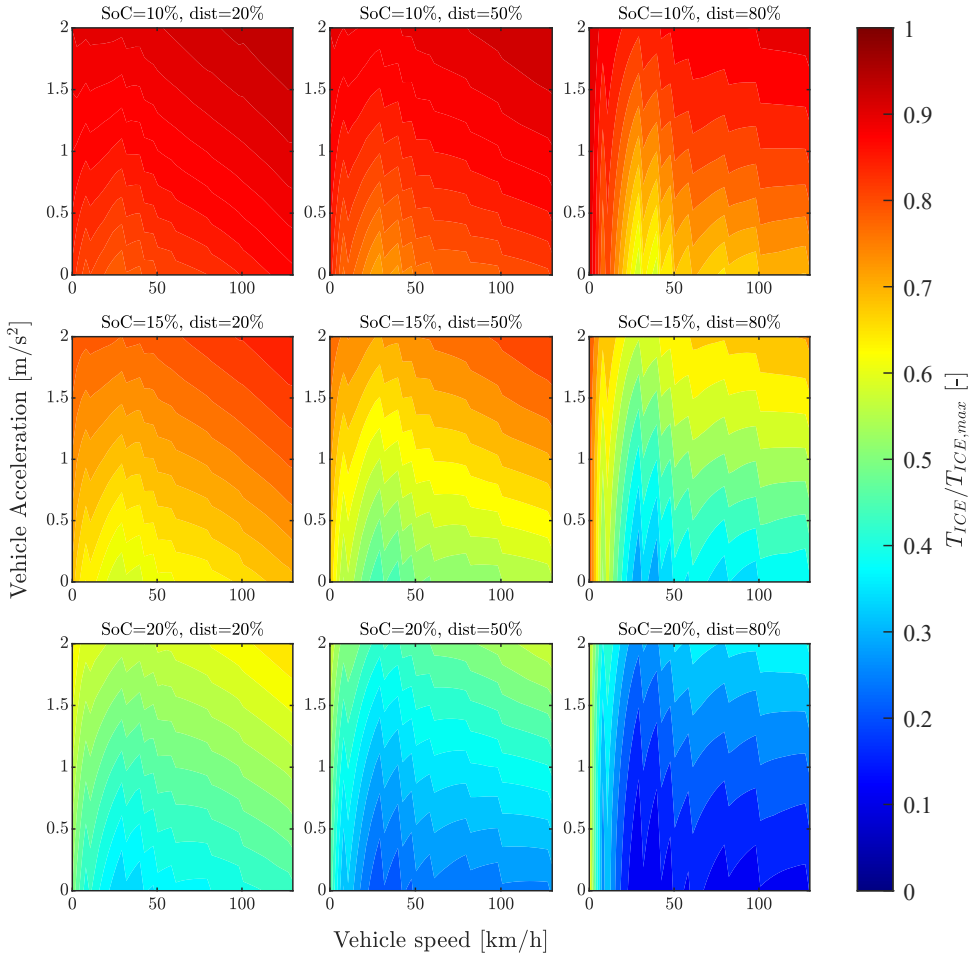


Fig. 3.30 Policy interpretability maps for the recurrent SAC agent. Normalized ICE torque command $T_{\text{ICE}}/T_{\text{ICE,max}}$ over a speed–acceleration grid for nine combinations of battery SoC (rows) and normalized travelled distance (columns). Warmer colors denote greater engine utilization.

3.4.4 Charge Depleting Policy

This section presents the CD policy results for the pHEV EMS, developed with the RL methodology of Section 3.4.2. To avoid degenerate learning and ensure the agent experiences meaningful engine–battery coordination, the driving-cycle dataset was filtered to exclude cycles that are trivially solvable in EV-only mode, i.e., missions for which the initial battery energy would allow a cycle termination without ICE support. Unlike CS, the CD objective tracks a non stationary SoC reference, defined as a linear discharge trajectory $SoC_{\mathrm{trg,CD}}$ parameterised by the travelled distance (see (3.51)). Accordingly, the analysis also examines the effect of supplying the agent with SoC target feedback as an additional state. The narrative proceeds as follows: baseline SAC training under CD operation is first characterized by analyzing reward evolution together with the joint trajectories of SoC and fuel to establish convergence and stability; the learned policy is then unboxed via the same interpretability protocol used in the CS section, by sweeping speed, acceleration, SoC, and trip progress, and both the control action (normalized ICE torque) and the discrete operating mode (EV, E-boost, load-point moving) are reported; subsequently, the impact of SoC target feedback is quantified by comparing a baseline agent with a variant augmented with SoC target feedback as additinal state, and recurrence is investigated by parameterizing the policy and value functions with an LSTM; finally, the controllers are validated on a high-fidelity forward virtual test rig, assessing tracking of the linear SoC target, terminal SoC compliance, and fuel/CO_2 outcomes, and contextualizing performance against reference strategies. Because CD introduces a dual out-of-distribution dimension, both the driving conditions and the initial battery energy, closed-loop performance is evaluated at three initial SoC set points (50%, 70%, 90%) to probe robustness across initial charge levels.

Baseline Policy Training Assessment

The learning dynamics of the CD controller are first assessed through the episode return in Figure 3.31. After a short transient with a few negative-return episodes, the mean reward rises steeply within the first 50–100 episodes and then exhibits a gentler upward drift before settling on a clear plateau from roughly episode 400 onward. The shaded band, representing the episode-wise standard deviation, is wide at the beginning, consistent with exploratory actions and cycle level stochasticity,

but progressively narrows, indicating improved policy consistency. The absence of late-stage drift or variance growth points to stable convergence toward a robust controller.

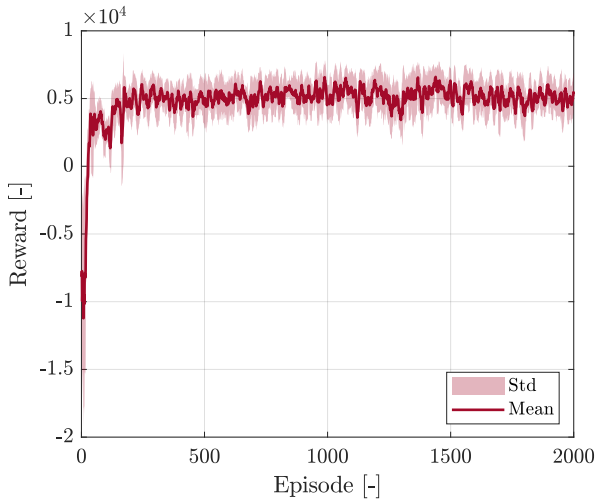


Fig. 3.31 Training loss for the CD SAC policy as a function of episodes. The solid line is the mean value across 20 episodes, while the shaded area is the standard deviation across 20 episodes.

Complementary insight is provided by the per-episode trajectories of SoC and cumulative fuel consumption in Figure 3.32 (colour-coded from early to late episodes). Despite the widespread of initial SoC values, all trajectories exhibit a controlled depletion and converge to a common terminal band around the final SoC target, demonstrating reliable enforcement of the final SoC constraint across episodes. As training progresses, dispersion near the end of the cycle shrinks markedly, confirming improved terminal tracking of the linear SoC reference. In parallel, cumulative fuel consumption evolves from irregular, episode-dependent growth in early training to smoother, clustered profiles in later episodes, consistent with a stabilized strategy that coordinates battery depletion and ICE usage.

Taken together, these results show that the CD agent not only converges in terms of reward, but also learns a physically coherent depletion policy that respects the terminal SoC target, irrespective of the starting SoC, while progressively regularizing fuel consumption behaviour.

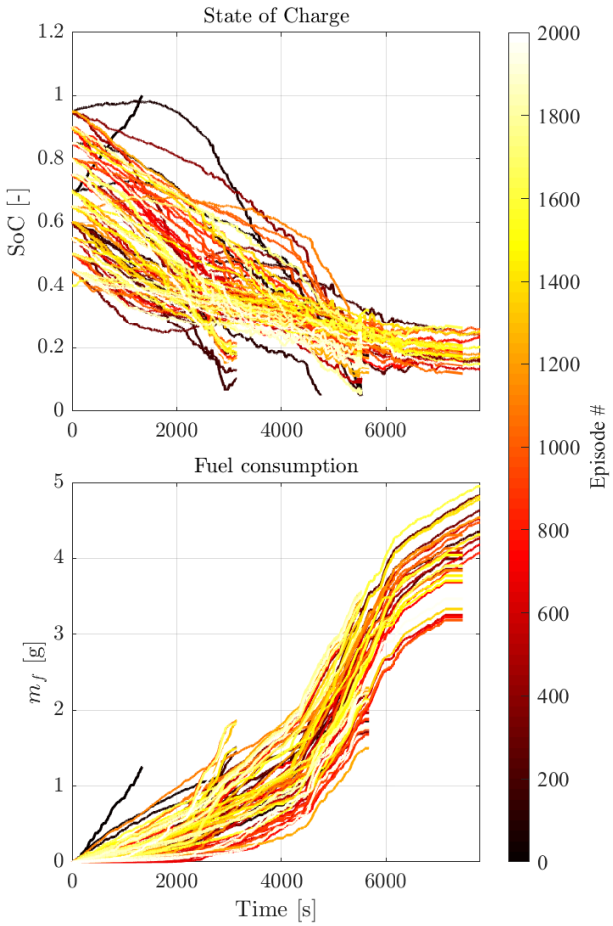


Fig. 3.32 Training evolution of the CD policy. Top: battery SoC versus time; bottom: cumulative fuel mass m_f versus time. Each curve corresponds to one episode and is colour–coded by episode index (dark = early, bright = late).

Decision-Making Process

As in the CS analysis, a post-hoc interpretability study is carried out to "unbox" the policy learned for the CD condition. A key difference with respect to CS is that the SoC levels explored in the maps vary with trip progress. This choice reflects the operational reality of CD deployment: the minimum SoC considered is 50%, so very low SoC values rarely occur at the beginning of a mission. Accordingly, the three SoC levels decrease with normalized travelled distance, as listed in Table 3.10. The policy is interrogated on a structured grid of vehicle speed, longitudinal acceleration, battery SoC, and trip progress; the resulting control action maps (normalized ICE torque) are reported in Figure 3.33, while the corresponding mode maps (EV, E-boost, load-point moving) are shown in Figure 3.34.

Table 3.10 Combinations of battery SoC and normalized travelled distance used for the policy-interpretability analysis under CD operation.

Normalized traveled distance	SoC level
20%	60%
	80%
	95%
50%	30%
	40%
	50%
80%	10%
	15%
	20%

A key outcome of this analysis is that, unlike CS, trip progress plays a leading role in the CD policy's decisions. Because the CD objective is tied to a distance-parameterized linear SoC target, the controller must coordinate the available electric energy with the remaining distance to track the desired depletion profile, which reflects the global optimum proposed by the DP.

Concering the control action (Figure 3.33), at 20% distance (left column), where the available electric energy is still high, the maps are almost uniformly blue, indicating EV-only operation across the domain to follow the desired depletion schedule. At 50% distance (middle column) with mid SoC levels (30–50%), the policy remains largely EV but begins to admit ICE torque in a bounded region at higher

accelerations and moderate to high speeds, highlighting a conditional relaxation when power demand rises or when SoC approaches the target. By 80% distance (right column), with low SoC (10–20%), the activation threshold drops markedly and ICE usage becomes widespread, especially for the 10–15% cases, ensuring drivability while preventing SoC from undershooting the lower bound before the trip ends. The systematic left-to-right shift, from EV dominance early to increasing ICE support late, demonstrates that the agent explicitly uses remaining distance to schedule depletion in line with the non stationary, distance-parameterized SoC target; this strong dependence contrasts with the CS logic, where distance had only a marginal effect.

Consistent with the control action maps, the mode-selection plots confirm that in CD operation, the policy is strongly organized by trip progress. At 20% distance (left column) with high SoC (60–95%), the maps are uniformly green: the controller runs EV mode everywhere to adhere to the early portion of the linear SoC target. By 50% distance (middle column) and mid SoC (30–50%), EV still dominates, but E-boost appears in the high-demand region (higher accelerations, moderate–high speeds), matching the bounded non-zero torque region seen previously; load-point moving remains negligible. At 80% distance (right column) with low SoC (10–20%), the balance shifts: E-boost becomes prevalent to guarantee performance, while load-point moving emerges at low accelerations and higher speeds to guard against undershooting the terminal SoC bound when the remaining distance is small. The left-to-right transition, EV early, selective E-boost mid-trip, and increased ICE involvement late, mirrors the torque-map trends and highlights the crucial role of remaining distance in scheduling depletion under the CD objective, in clear contrast to the marginal distance effect observed in CS.

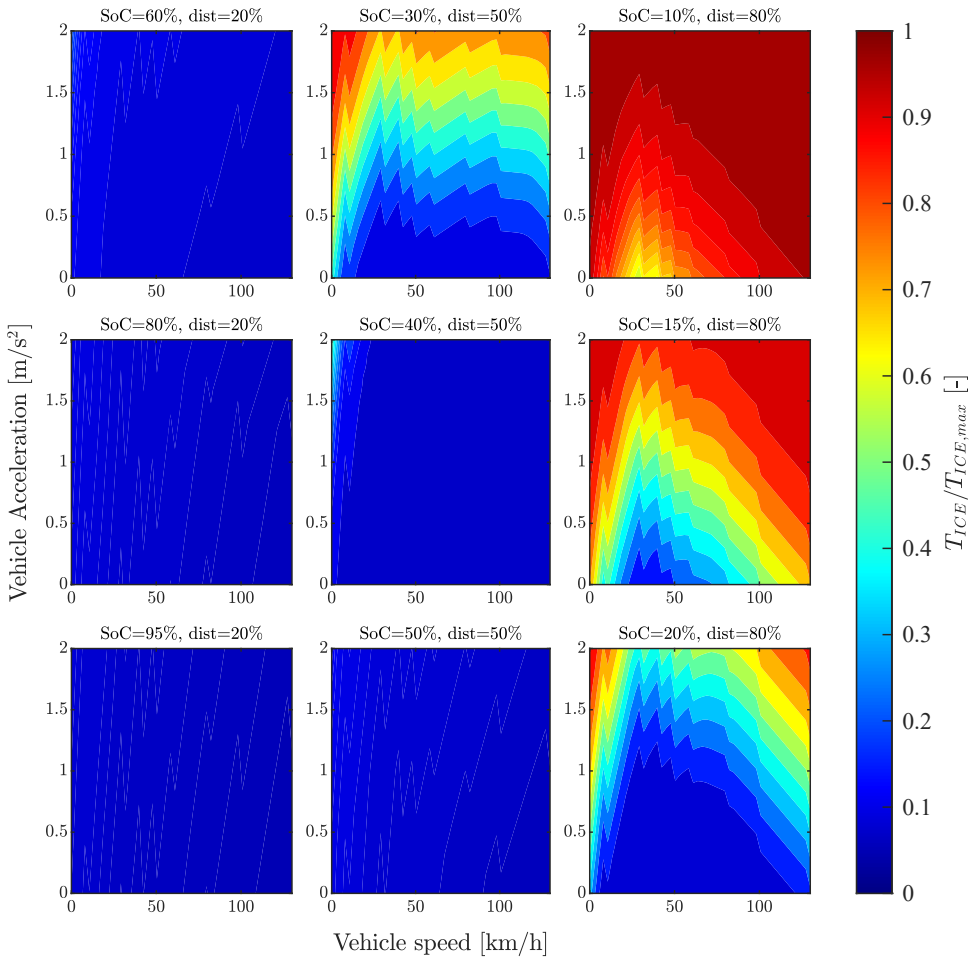


Fig. 3.33 Grid of normalized action distributions for nine combinations of SoC and travelled distance percentage; each panel shows the normalized engine torque within its corresponding SoC-distance bin for the CD logic.

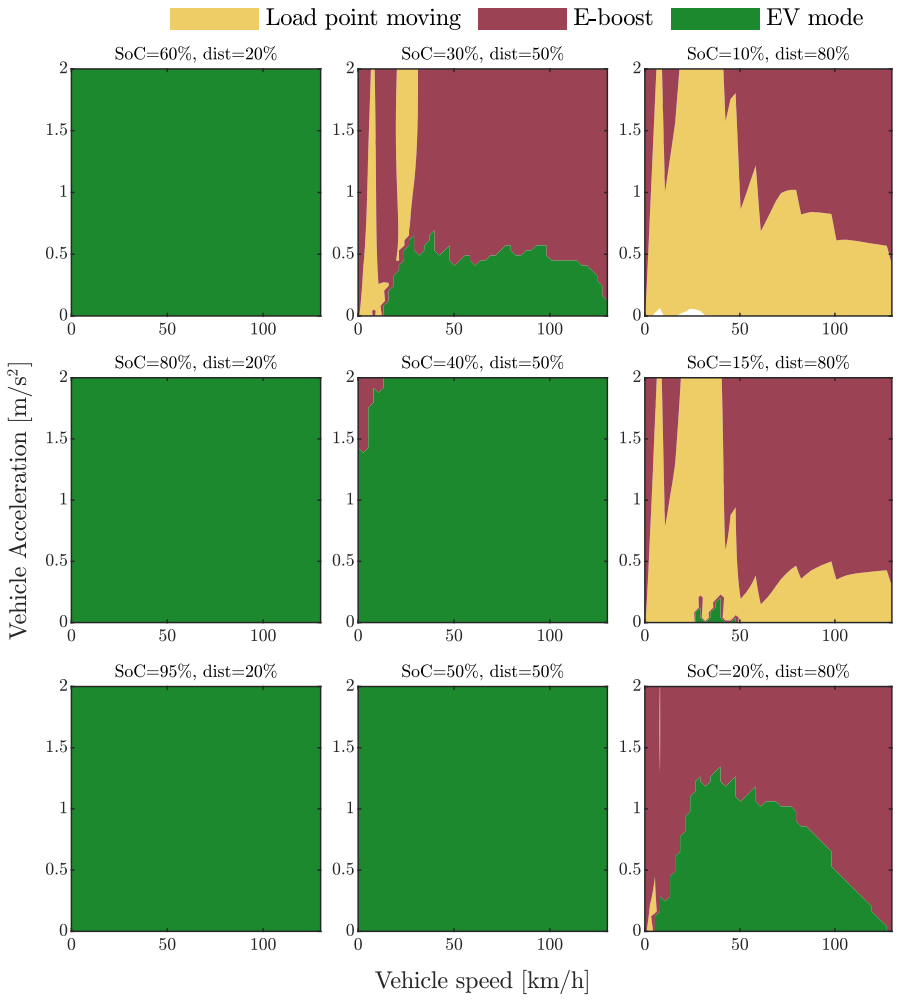


Fig. 3.34 Grid of speed–acceleration plots for nine combinations of SoC and travelled distance percentage; each panel shows the engine management strategies within its respective SoC and distance interval for the CD logic.

Effect of SoC Target Feedback

Trip distance strongly influences CD decision-making, which is critical to fully exploit the benefits of pHEVs. A desirable strategy yields approximately linear battery SoC depletion with respect to traveled distance, rather than relying on pure-EV operation and subsequently switching to CS mode at low SoC, where battery efficiency deteriorates. Consequently, effective fuel-minimizing control requires at least the remaining trip distance, information that is easily available from modern GPS systems.

To enhance the observability of the MDP, the state is augmented with a distance-parameterized reference SoC trajectory, denoted $SoC_{\text{trg,CD}}$, defined in (3.51). This reference is not constant, as in the CS operation, and defines a linearly decreasing profile with distance. The same quantity is used in the reward design, (3.52), to guide learning toward the desired CD behavior. The resulting policy, referred to as *SAC w* SoC_{trg}, is compared with a baseline SAC agent in terms of training stability and closed-loop control on the usual RDE driving cycle, using three initial SoC levels: 50%, 70%, and 90%.

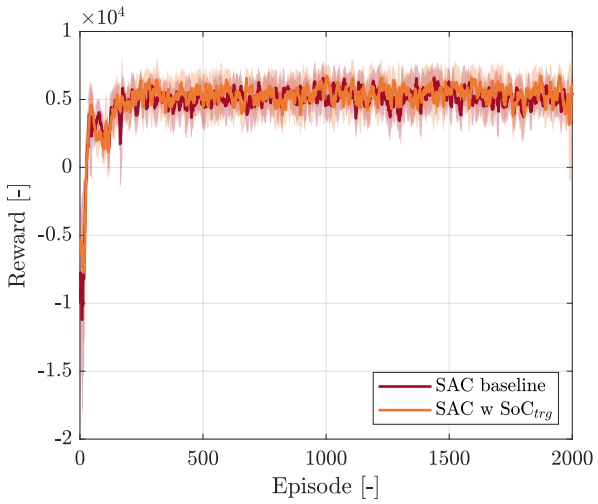
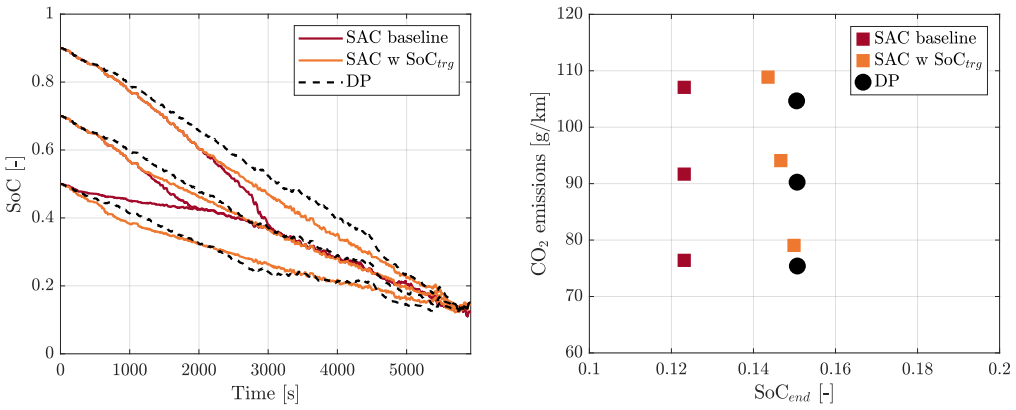


Fig. 3.35 Training reward versus episode for the SAC baseline (red) and the SAC agent augmented with the reference SoC state (orange).

Closed-loop behavior on the backward kinematic virtual test rig is summarized in Figure 3.36. In panel (a), SoC trajectories display the targeted near-linear depletion across all initial SoC conditions. The augmented policy tracks the distance-



(a) (b)

Fig. 3.36 Closed-loop evaluation on the backward kinematic virtual test rig over an RDE cycle for three initial SoC levels (50%, 70%, 90%). In (a), SoC trajectories are reported, while in (b), CO_2 emissions versus terminal SoC (SoC_{end}).

parameterized reference more closely and aligns better with the DP benchmark (black dashed), whereas the baseline shows greater deviation, manifested as EV-heavy operation early in the cycle at high initial SoC (90%) or increased ICE contribution at low initial SoC (50%). Panel (b) compares CO_2 emissions against terminal SoC (SoC_{end}). As expected, emissions increase with higher final SoC values. Both learned policies achieve emission levels close to DP across the three scenarios and close meet the terminal SoC target of 15%. Notably, the augmented policy attains a terminal SoC closer to the target with only a limited change in CO_2 emissions relative to the baseline. The combination of improved SoC-tracking throughout the drive and accurate terminal SoC regulation supports the effectiveness of including the reference trajectory as an additional state variable to further enhance the performance of the CD logic.

Effect of Recurrence

Following the same approach adopted for the CS policy (see Section 3.4.3), a natural way to improve MDP observability is to provide the agent with temporal memory via recurrent networks. To this end, an LSTM-based SAC agent (hereafter, *SAC recurrent*) was trained and compared against the baseline SAC using the usual

two-step protocol: (i) analysis of the training average reward and (ii) closed-loop evaluation on a common test cycle.

The training behavior is broadly similar across agents. As shown in Figure 3.37, the recurrent SAC (teal) reaches the stationary cumulative-reward level slightly faster, but the asymptotic value and the spread of the cumulative reward are comparable to those of the baseline (red). This indicates similar training stability and steady-state performance.

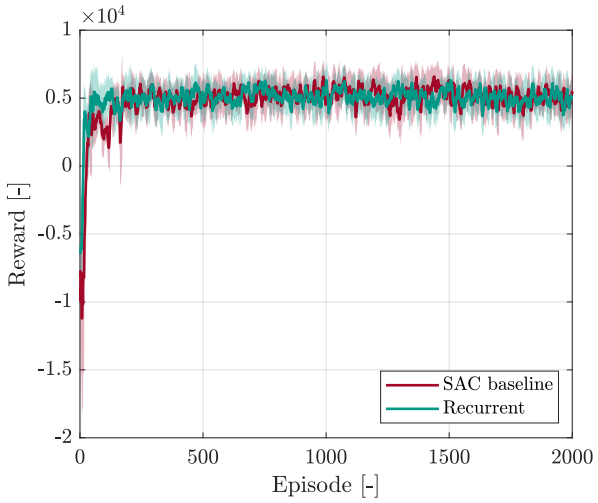
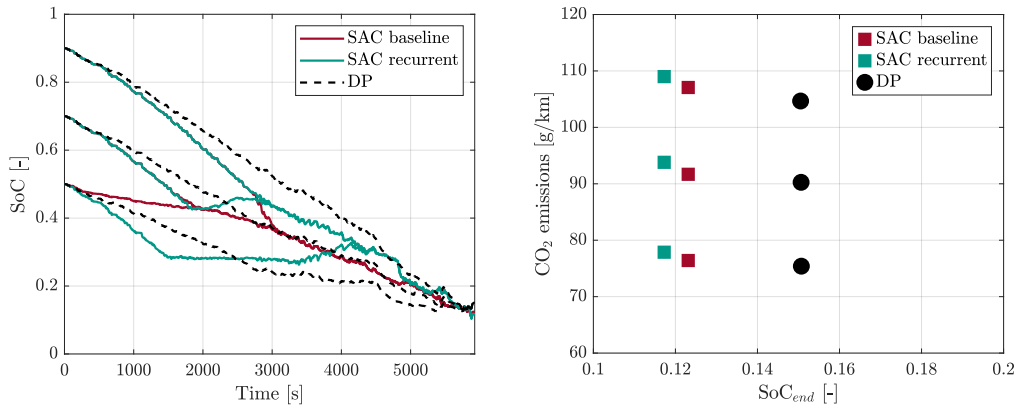


Fig. 3.37 Training reward versus episode for the SAC baseline (red) and the LSTM-based SAC recurrent agent (teal).

Closed-loop performance was assessed on the backward kinematic virtual test rig over the RDE cycle for three initial SoC conditions (50%, 70%, 90%), see Figure 3.38. In panel (a), both agents satisfy the terminal SoC constraint ($SoC_{\text{end}\approx 15\%}$) in all scenarios. The recurrent policy displays a pronounced two phase pattern, aggressive early depletion followed by a conservative mid-cycle segment and a late correction, yielding a more curved trajectory that is further from the approximately linear DP trend (black dashed). The baseline SAC shows a steadier, more DP-like depletion profile. Panel (b) reports CO_2 emissions versus the final SoC value. Both learned policies lie close to DP, but the recurrent agent is typically equal to or slightly higher than the baseline, with a final SoC value almost equal to the baseline, highlighting a slightly less efficient exploitation of the fuel chemical energy. In this setting, recurrence offers no clear efficiency advantage: constraint satisfaction is preserved, yet SoC evolution is less regular and emissions benefits are not evident.



(a) (b)

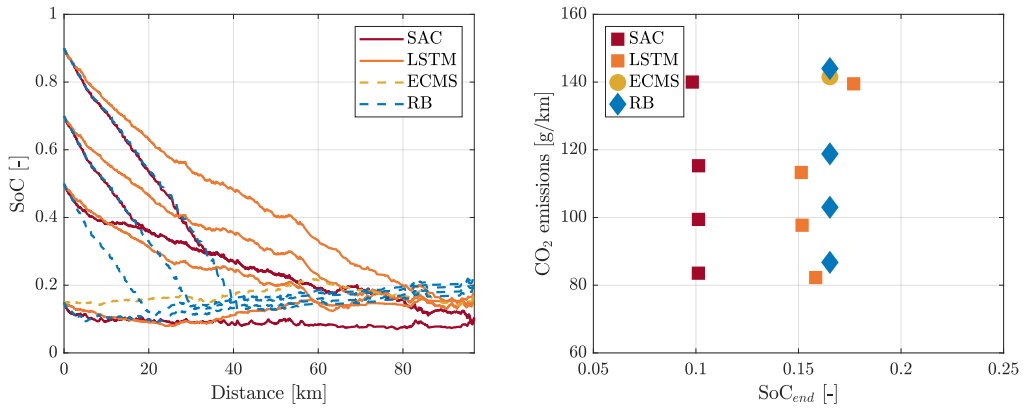
Fig. 3.38 Closed-loop evaluation on the backward kinematic virtual test rig over an RDE cycle for three initial SoC levels (50%, 70%, 90%). In (a), SoC trajectories are compared, while in (b), the trade-off between CO_2 emissions and final SoC value is presented.

3.4.5 Performance on Detailed Virtual Test Rig

To complement the backward-rig evaluation and validate the controllers under realistic transients and plant constraints, the learned policies were assessed on a high-fidelity forward dynamic virtual test rig developed via GT–Simulink co-simulation. Unlike the backward formulation, this environment explicitly accounts for actuator dynamics, driveline losses, and nonlinear interactions, thus providing a more stringent and physically consistent benchmark for EMS performance.

Figure 3.39 summarizes the closed-loop results. Panel (a) reports the SoC trajectories for four different initial values (15%, 50%, 70%, 90%). Both SAC and the LSTM-based controller regulate the battery in a smooth and adaptive manner: irrespective of the starting condition, the trajectories converge consistently toward the charge-sustaining target without violating the lower bound, while flexibly exploiting the battery buffer to limit ICE usage. It is worth noting that the SAC policy trained on the backward kinematic rig tends to discharge slightly deeper (final SoC around 11%), an effect mainly attributable to the higher instantaneous energy demand modeled in the forward simulation. In contrast, the RB controller displays a two-phase pattern: it initially depletes the battery almost entirely in EV mode—evident in the steep initial SoC drop—before switching to charge-sustaining operation. This sequential strategy is inherently inefficient, as it forces the ICE to operate more frequently in suboptimal regions once the electric buffer is exhausted. The ECMS, which is inherently defined in CS operation, maintains the SoC close to the target but tends to induce a slight overcharge toward the end of the cycle, reflecting its myopic instantaneous optimization principle.

Panel (b) quantifies the corresponding CO_2–SoC trade-off. SAC and LSTM consistently achieve lower emissions than both RB and ECMS while still converging to the desired terminal SoC. Between the two learning-based policies, SAC exhibits slightly deeper depletion, translating into marginally lower CO_2 values, whereas LSTM enforces smoother SoC regulation at the expense of a modest emission increase. ECMS, which was tested only in CS conditions, delivers competitive results but requires explicit parameter tuning, while RB is clearly dominated, combining higher emissions with larger deviations from the SoC target. Overall, the forward-rig results confirm the superior adaptability of the learned controllers, which outperform classical baselines both in aggregate performance and in the efficiency of their SoC management strategy.



(a) (b)

Fig. 3.39 Closed-loop evaluation on the forward dynamic virtual test rig over an RDE cycle for four initial SoC levels (15%, 50%, 70%, 90%). In (a), SoC trajectories are compared, while in (b), the trade-off between CO_2 emissions and final SoC value is presented.

A more detailed view is provided by the engine operating-point time distributions. In CS with a 15% SoC target (Figure 3.41), both SAC and LSTM concentrate operating time around the high-efficiency island of the Brake Specific Fuel Consumption (BSFC) map, confirming their capability to systematically shift the ICE toward favorable load–speed regions. This explains the smoother SoC regulation and lower emissions observed in Figure 3.39. ECMS also exploits efficient regions, but its operation is more narrowly clustered around specific load points, a direct consequence of its instantaneous optimization principle. By contrast, RB scatters operating points over a wider area, mainly at low loads, where engine efficiency is poor. This dispersion is consistent with its two-phase SoC behavior (initial EV depletion followed by charge-sustaining operation), which forces the ICE to operate frequently in suboptimal conditions and results in higher fuel consumption.

In CD starting from a high initial SoC of 90% (Figure 3.41a), the learning-based controllers exploit the battery aggressively, but when the ICE is requested they employ it sparsely at *high load* and near the BSFC valley (i.e., load-point–moving operation), explicitly avoiding low-load, inefficient points; this mechanism explains the SoC trajectories in Figure 3.39a and the lower CO_2 levels in Figure 3.39. Conversely, RB switches to a sustaining phase after its initial EV-only discharge and activates the ICE across a much wider area—including many low-load regions—thereby operating

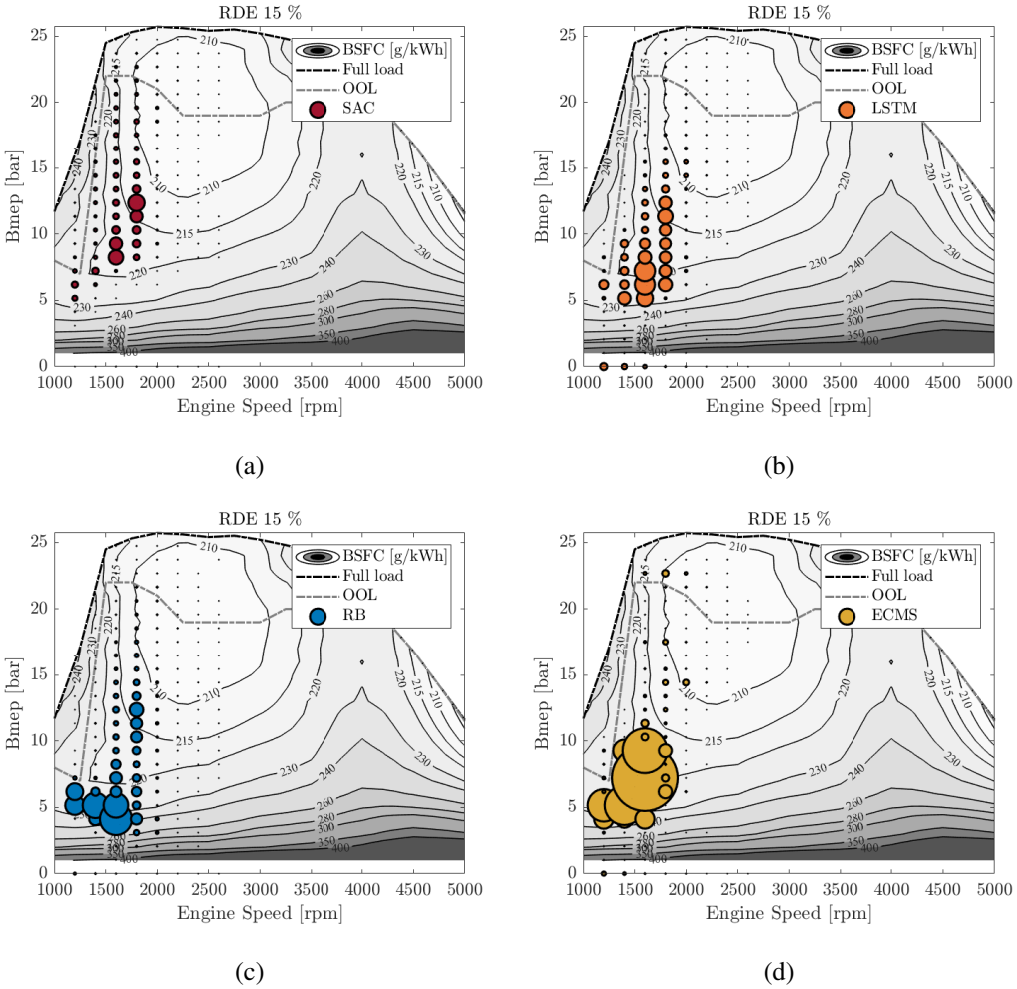


Fig. 3.40 Engine operating point time distribution in CS conditions with a SoC target of 15% for different control strategies: (a) SAC, (b) LSTM, (c) RB, and (d) ECMS. The background shows the BSFC map of the case study engine.

away from the optimal efficiency locus and incurring the higher emissions observed in the SoC–CO_2 trade-off.

$$CO_{2,\mathrm{eq}} = \frac{\left(SoC_{\mathrm{end}} - SoC_{\mathrm{trg}}\right) E_{\mathrm{batt}}}{d_{\mathrm{veh}}} \frac{BSCO_{2,\mathrm{avg}}}{\eta_{\mathrm{path}}} \tag{3.53}$$

Since the forward dynamic evaluation does not strictly enforce a perfect SoC target, the final SoC may slightly deviate from the target value. To ensure a fair comparison among strategies, especially across different initial SoC levels, the emissions are therefore corrected according to Equation (3.53).

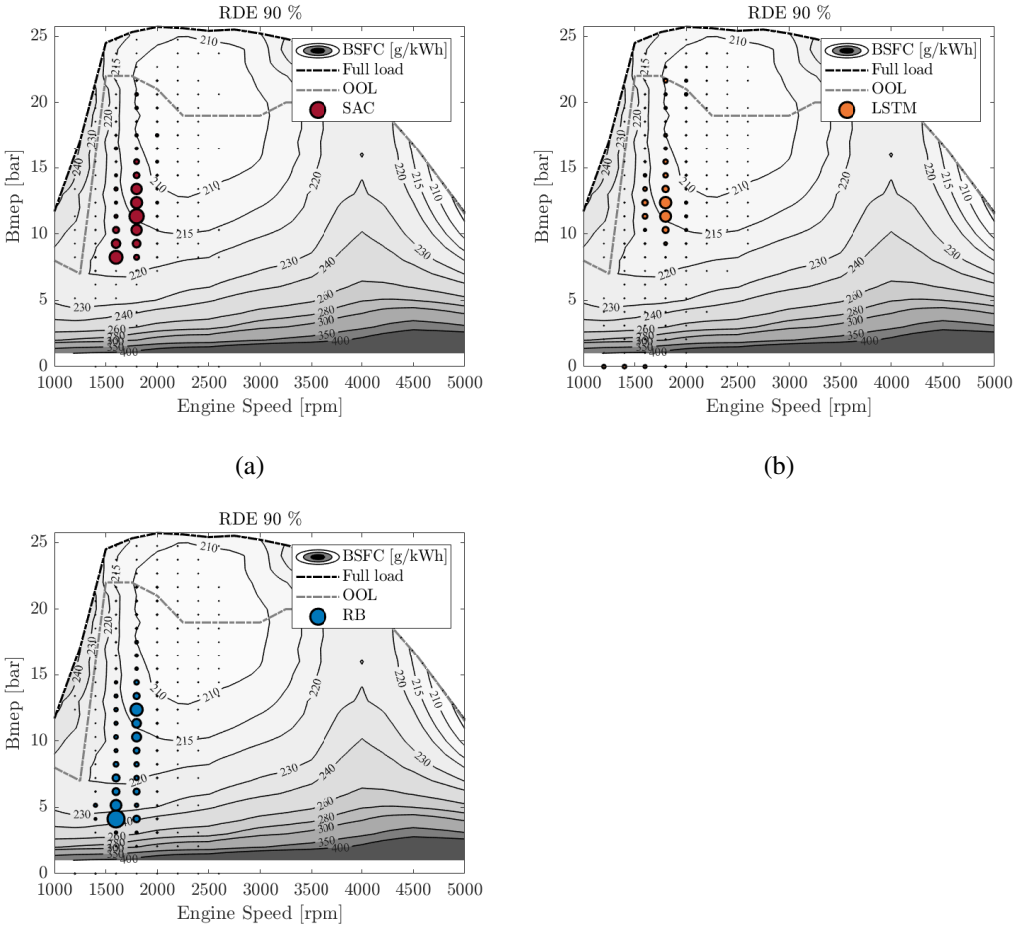


Fig. 3.41 Engine operating point time distribution in CD conditions starting from a SoC of 90% for different control strategies: (a) SAC, (b) LSTM, and (c) RB. The background shows the BSFC map of the case study engine.

The formulation converts the residual battery energy imbalance into an equivalent CO_2 contribution by means of the average engine specific CO_2 emissions and a global energy conversion efficiency between the engine shaft and the battery. In this way, any under- or over-depletion of the battery buffer is consistently translated into an equivalent fuel penalty (or credit), thus restoring energetic comparability across controllers.

Table 3.11 reports the resulting equivalent CO_2 emissions for all control strategies and initial SoC levels. The values include both the measured tailpipe emissions and the SoC-based correction term defined in Equation (3.53), thus ensuring a fair

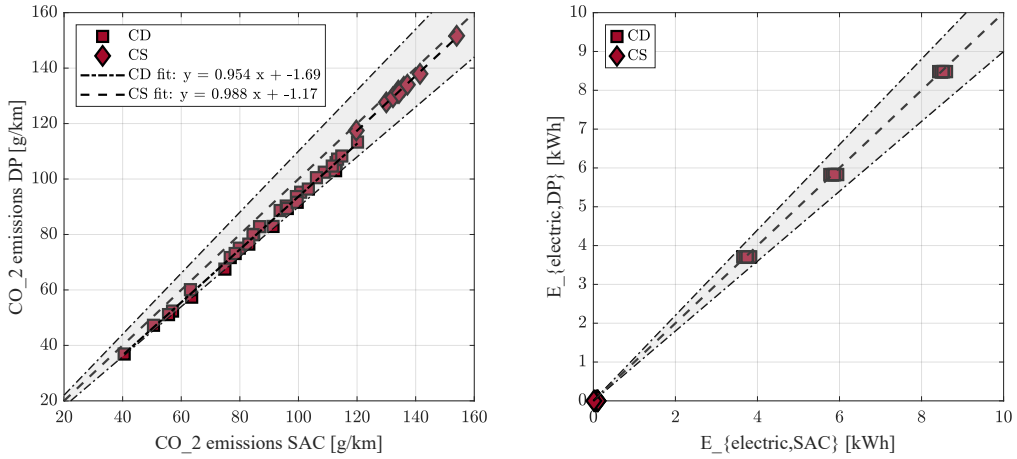
Table 3.11 Equivalent CO_2 emissions including SOC correction for different control strategies and initial SOC levels.

	SOC 15% (%Dev. RB)	SOC 50% (%Dev. RB)	SOC 70% (%Dev. RB)	SOC 90% (%Dev. RB)
Rule-Based	144.46 (0%)	119.44 (0%)	103.60 (0%)	87.11 (0%)
ECMS	141.67 (−1.9%)	–	–	–
LSTM	138.21 (−4.3%)	113.31 (−5.1%)	97.73 (−5.7%)	82.33 (−5.5%)
SAC	138.60 (−4.1%)	112.48 (−5.8%)	96.90 (−6.5%)	81.44 (−6.5%)

comparison across controllers that lead to a different final battery energy content. Consistently with the trends observed in the SoC–CO_2 trade-off of Figure 3.39, the learning-based controllers achieve the lowest emissions for all initial SoC levels. Both SAC and LSTM systematically outperform the RB strategy, with reductions that increase as the initial SoC rises. At low initial SoC (15%), LSTM and SAC exhibit comparable performance, with LSTM marginally outperforming SAC. However, at medium and high initial SoC levels (50–90%), SAC achieves the lowest equivalent CO_2 emissions, reaching reductions of up to 6.5% relative to the RB benchmark. This trend is consistent with the more aggressive yet structured load-point–moving behavior observed in the CD operating point distributions, where SAC more effectively exploits high-efficiency ICE regions while avoiding low-load operation. ECMS was evaluated exclusively under CS conditions, and therefore, its comparison with the learning-based controllers is restricted to this operating regime. Within CS operation, ECMS delivers competitive results, confirming the effectiveness of its instantaneous optimization principle in maintaining the SoC close to the target while exploiting relatively efficient ICE operating regions. Nevertheless, both SAC and LSTM achieve lower equivalent CO_2 emissions even under CS constraints, indicating that their learned policies are able to implicitly capture the long-term impact of operating-point selection beyond instantaneous fuel optimality.

Overall, the results obtained on the detailed forward dynamic test rig confirm that the SAC agent and LSTM model outperform the RB and ECMS baselines in terms of SoC management and CO_2 emissions, while also demonstrating structured and interpretable operating patterns aligned with efficient ICE usage. Nevertheless, the observed slight SoC under-discharge of SAC in CS conditions and the small residual

deviations from the optimal CO_2–SoC trade-off suggest that additional performance gains could be unlocked through a targeted fine-tuning of the reward formulation and hyperparameters directly on the forward rig. Such calibration, accounting for the higher-fidelity dynamics of the co-simulation environment, would enable the SAC policy to better adapt to real plant characteristics and further narrow the gap to the optimal benchmark.



(a) (b)

Fig. 3.42 Comparison between SAC and DP results: (a) CO_2 emissions [g/km] and (b) electric energy consumption [kWh]. Results are shown for CD and CD conditions. Dashed lines indicate ±10% deviation.

3.4.6 Generalization Performance

The previous sections established (i) the sample efficiency and performance advantages of SAC over DDQL, (ii) the structure of the learned decision mechanisms under CS and CD operation, (iii) the limited role of distance for CS and its centrality for CD, (iv) the marginal gains of recurrent parametrizations, and (v) the closed-loop validity of the policies on a high-fidelity forward rig. We conclude by assessing *generalization*: does the baseline SAC reproduce the behavior of the optimal benchmark (DP) across unseen and very diverse driving conditions and different initial battery states?

To answer this, we evaluated SAC and DP on a test matrix comprising 10 heterogeneous driving cycles and 4 initial SoC levels (90%, 70%, 50%, and 15%) on the backward kinematic virtual test rig to be compliant with the training environment and do not introduce biases due to the different environment fidelity level. Figure 3.42 summarizes the results. Each marker corresponds to one cycle–SoC combination, with symbols distinguishing CD and CS operation; the 1:1 line and the dashed 10% bands provide visual error bounds, and the shaded area highlights the agreement region.

Panel 3.42a shows DP (x-axis) versus SAC (y-axis) tailpipe CO_2 emissions. The cloud of points is tightly concentrated around the identity line and largely contained within the 10% error corridor, indicating an overall good agreement between SAC-based and DP-based CO_2 emission estimates. However, a more detailed inspection reveals a regression trend with a slope slightly lower than unity, suggesting a systematic tendency of the SAC approach to yield higher CO_2 emissions than the corresponding DP values. This effect is particularly evident under CD operation, where most data points lie above the 1:1 line, resulting in a regression trend that deviates more from the ideal identity relationship than in CD operation (see Panel 3.42a). This deviation is therefore indicative of a consistent bias rather than random scatter. This discrepancy is an expected consequence of the different optimization information structures. DP provides a benchmark solution because it optimizes with full knowledge of the entire driving profile, whereas SAC operates online and bases its decisions on the current state, using only the remaining running distance as limited future information. As a result, reproducing DP CO_2 emissions exactly is not achievable in principle. The objective of the proposed SAC formulation is instead to reduce the gap with respect to the DP benchmark as much as possible while guaranteeing satisfaction of the final SoC target. Despite this inherent limitation, the results demonstrate that SAC matches the DP outcome with high fidelity across operating regimes (CD and CS), driving cycles, and initial SoC conditions. The close clustering around the identity line, together with the largely uniform containment within the 10% corridor, confirms that SAC delivers consistently near-optimal CO_2 performance over a wide operating range, with only a modest and systematic offset that remains bounded even at higher emission levels.

guaranteeing fulfillment of the final SoC target.

Panel 3.42b reports the correlation in total electric energy usage. Again, the points cluster near the identity line and stay mostly within the error band. The absence of a visible slope or offset error suggests that SAC closely matches DP's *allocation* of electric energy versus fuel, rather than simply trading one for the other. In other words, SAC has learned a close to DP global scheduling principles, despite operating without preview and under partial observability during training. To conclude, this last section proves that the SAC controller generalizes very well across 10 different driving cycles and 4 different initial SoC levels, i.e., 40 different test conditions. It reproduces DP's CO_2 outcomes and total energy usage with strong correlation and bounded deviations. Combined with the interpretability analysis and

forward-rig validation, these results indicate that the learned policy has captured transferable control structure (mode selection and ICE loading) rather than overfitting to specific cycles or initial conditions. As such, SAC provides a practical, preview-free approximation of the DP benchmark suitable for real-time EMS deployment.

3.5 Conclusion and Future Steps

This chapter demonstrated a unified, learning-centric control framework operating across two scales. At the network level, a hierarchical SAC–GNN policy for AMoD coordination achieved near-MPC operator profit while delivering the lowest waiting times among feasible baselines, and did so with disciplined rebalancing that moderated distance and emissions. The policy generalized well across spatial aggregations, time-of-day regimes, and simulator fidelities; macroscopic pretraining transferred effectively to mesoscopic evaluation, and the reward weight β offered a single interpretable lever to navigate cost–service trade-offs. At the powertrain level, the proposed EMSs—DP-informed LSTM and preview-free SAC produced structured, physically consistent decisions (EV, boost, load-point shifting), matched DP trends on CO_2 and electric-energy usage across heterogeneous cycles and initial SoC levels, and outperformed RB and ECMS baselines on a high-fidelity forward rig. In CS, recurrence brought marginal gains versus a simpler SAC; in CD, augmenting the state with a distance-parameterized SoC target improved tracking without sacrificing efficiency.

Future work will explore extending these methods to more complex multi-agent scenarios, integrating real-time data streams (e.g., disruptions and demand adaptations), and optimizing for additional objectives, such as energy efficiency (e.g., for electric vehicles) and equity in service distribution. For the hybrid power-split scale, priorities include a systematic exploration of recurrent architectures with targeted hyper-parameter optimization, the integration of a short-horizon speed (or load) forecasting layer to increase effective observability, and richer learning signals, e.g., offline RL for data-efficient policy improvement and meta-learning to accelerate adaptation and strengthen generalization across vehicles, cycles, and operating contexts.

Chapter 4

Modelling Application to Predictive Combustion Models

Combustion modeling in ICEs shares the same systems-control pressures highlighted at the other scales in this dissertation: decisions must be accurate, fast, and robust to uncertainty, especially for real-time applicationsation, like the implementation of combustion models in real vehicle ECUs. Here, the trade-off is between (i) high-fidelity physical descriptions, up to CFD, that resolve turbulence-chemistry interaction but remain too expensive for large DoE campaigns, digital twins, and ECU embedding, and (ii) compact empirical surrogates that are computationally attractive yet can lose accuracy, and have a limited applicability close to calibration operating conditions. Real-time applications demand a middle path that preserves accuracy and physical trends while delivering fast inference.

The chapter positions two complementary families within the standard 0D/1D thermodynamic backbone. Phenomenological models encode premixed-flame physics via laminar burning velocity and turbulence-induced surface amplification in single- and two-zone settings. Data-driven models, by contrast, learn the mapping from operating conditions to crank angle–resolved combustion descriptors. These surrogates are introduced not as classical black-box replacements but as physics-aware complements: feature selection and loss shaping reflect combustion causality; outputs interface directly with single-/two-zone balances so that predicted burn rate profiles yield pressure and heat release histories without violating the main physical rules.

The central motivation is practical: approach the accuracy of physically and phenomenologically informed models where fixed empirical models struggle, while retaining the computational efficiency required for calibration loops, virtual sensing, and closed-loop control. In this role, data-driven models act as fidelity amplifiers for 0D/1D workflows, extending expressiveness beyond strictly S-shaped MFB assumptions yet maintaining portability and real-time performance across operating points.

This chapter is structured in the following way. First, Section 4.1 reviews the phenomenological models for SI combustion in 0D/1D and indicates where empirical forms typically lose accuracy. Section 4.2 presents the PHOENICE engine and dataset. Section 4.3 details the data-driven surrogates, an NN-parameterized Wiebe for ultra-fast reconstruction, and a hybrid sequence model that predicts MFB over crank angle with physics-informed regularization. Section 4.4 evaluates accuracy and generalization across speed, load, and dilution using MFB10, MFB50, and MFB10–75 alongside full burn rate profiles. Section 4.5 discusses implications for real-time deployment and outlines co-simulation and dataset-enrichment steps.

4.1 Combustion Models

This section outlines the modeling strategies used to characterize combustion in ICE with an emphasis on Spark Ignition (SI) operation and 0D/1D practice. Our focus is on the quantities that drive system analysis and control, namely the burn rate and heat release profiles over crank angle, and their scalar summaries such as CA10/50/90, burn duration, and peak pressure. Two complementary families are reviewed. Phenomenological models retain a compact, physics-based description of premixed flame propagation: local chemistry enters through laminar burning-velocity correlations, while turbulence augments flame surface, yielding practical closures that are interpretable but require calibration. Data-driven models learn the mapping from operating conditions and controls to combustion descriptors directly from data, trading explicit turbulence/chemistry closures for speed and adaptability while remaining anchored to the same thermodynamic framework. The remainder proceeds as follows. Section 4.1.1 introduces laminar and turbulent premixed-flame fundamentals and reviews mainstream closures in single- and two-zone settings (Wiebe, eddy burn-up, fractal, CFM). Section 4.1.2 then surveys recent data-driven

methods and presents the LSTM sequence model used in this work, which predicts the full crank angle–resolved burn rate at low computational cost.

4.1.1 Phenomenological

Phenomenological combustion models aim to reproduce the leading-order physics of SI burning with minimal computational cost and a small, interpretable set of parameters. They idealize the flame as a thin front propagating through a premixed charge and separate two roles: local chemistry, encapsulated by the laminar burning velocity and flow-induced enhancement, introduced through turbulence metrics that wrinkle and stretch the front, increasing its effective area. In 0D/1D engine simulations, these models are embedded in single-zone or two-zone thermodynamic frameworks and are coupled to geometry-based volume laws and wall heat-transfer correlations. The objective is to deliver a credible Mass Fraction Burned (MFB), or, equivalently, the heat-release history across speed, load, dilution, and spark sweeps, suitable for system studies, calibration, and emissions/knock hooks, while remaining transparent about assumptions and tuning needs. The subsection proceeds from laminar flame fundamentals to turbulent flame phenomenology, then reviews mainstream closures used in practice: a single-zone baseline with Wiebe MFB, and two-zone formulations in which the burn rate is set using three different submodels.

Laminar premixed flames. A premixed flame arises when fuel and oxidizer are mixed before ignition. The flame front is a thin, self-sustaining reaction layer that separates unburnt and burnt gases and propagates into the fresh mixture; peak temperatures can exceed 2000 K. Depending on the surrounding flow, premixed flames are classified as laminar or turbulent.

In the laminar case, the internal structure is classically decomposed into a relatively thick preheat zone, dominated by species and heat diffusion with negligible heat release, followed by a thin reaction zone with intense exothermic chemistry. The laminar burning velocity u_L, which represents the normal propagation speed into the fresh gas, is an intrinsic indicator of mixture reactivity. Balancing transport and reaction rates yields the familiar scalings

$$u_L \propto \sqrt{\frac{\lambda \omega_r}{p}} \tag{4.1}$$

$$\delta_L \propto \frac{\lambda}{{}_p u_L} \tag{4.2}$$

with λ thermal conductivity, ω_r reaction rate, ρ density, c_p specific heat, and δ_L laminar flame thickness. The preheat layer is typically an order of magnitude thicker than the reaction layer, where the structure of a premixed laminar flame is reported in Figure 4.1.

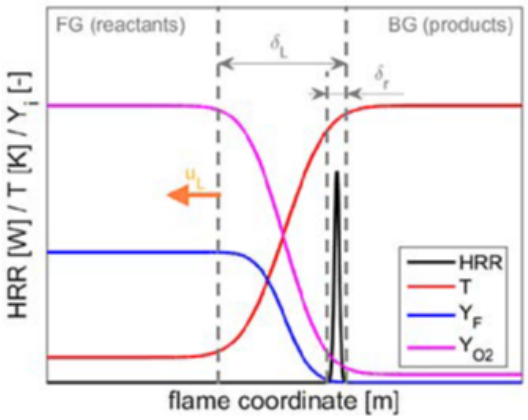


Fig. 4.1 Structure of a laminar premixed flame [3].

The laminar burning velocity u_L depends strongly on equivalence ratio ϕ, unburnt temperature T, pressure p, and dilution (Exhaust Gas Recirculation (EGR)/residuals). For gasoline-like fuels, u_L peaks for slightly rich mixtures ($\phi \approx 1.05$–1.15), decreases toward the flammability limits, increases with higher T (faster chemistry and higher conductivity despite lower density), and decreases with p (diffusive transport within the front is hindered even as reactant concentration rises). Dilution slows the flame because added heat capacity lowers the reaction-zone temperature (Figure 4.2). For SI-engine modeling, these trends are commonly captured by power-law correlations referenced to (T_0, p_0) [150] and extended for dilution:

$$u_L = u_{L,0}\left(\frac{T}{T_0}\right)^{\alpha(\phi)}\left(\frac{p}{p_0}\right)^{\beta(\phi)} \tag{4.3}$$

$$u_L^{(\mathrm{dil})} = u_L[1 - GX_{\mathrm{dil}}]^{\gamma} \tag{4.4}$$

where $u_{L,0}$, α, β, G, and γ depend on fuel formulation and X_{dil} is the molar fraction of diluent.

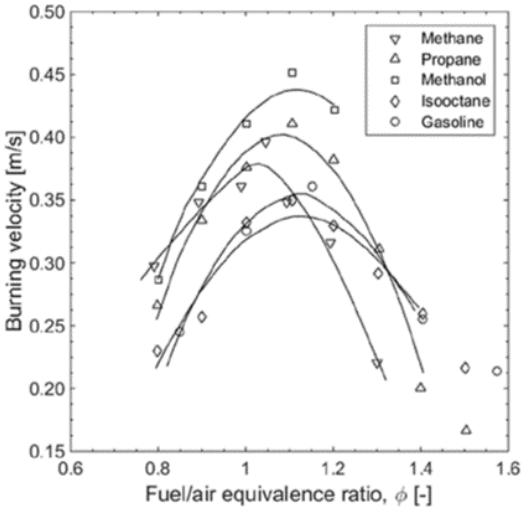


Fig. 4.2 Laminar burning velocities of selected fuels versus equivalence ratio at 0.1 MPa and 300 K. Lines are leastsquares polynomial fits to experimental data [4].

While the inner micro-structure of the front is laminar, engine flows are rarely quiescent. During the main burn the in-cylinder motion corrugates the front over a range of scales, so that chemistry remains locally laminar-like but the effective burning rate is governed by how turbulence increases flame surface. This motivates the following turbulence metrics and the associated regime diagram.

Turbulent premixed flames. In turbulent premixed combustion, the flame front interacts with eddies across a spectrum of scales; turbulence predominantly wrinkles and stretches the front, increasing surface area and thus the global burning rate. The phenomenology is organized by three non-dimensional groups: the turbulent Reynolds number Re_t, which represents the ratio of inertial to viscous effects at the integral scale; the Damköhler number Da, that is the ratio of large eddy turnover time to chemical flame time; and the Karlovitz number Ka, which is the ratio of chemical

time to Kolmogorov time, indicating how fast the smallest eddies act relative to chemistry:

$$\begin{aligned} \mathrm{Re}_t &= \frac{u' l_t}{\nu} \\ \mathrm{Da} &= \frac{\tau_{l_t}}{\tau_{\mathrm{ch}}} = \frac{l_t/u'}{\delta_L/u_L} = \frac{u_L l_t}{u' \delta_L} \\ \mathrm{Ka} &= \frac{\tau_{\mathrm{ch}}}{\tau_\eta} = \frac{\delta_L/u_L}{\tau_\eta} \end{aligned} \tag{4.5}$$

where u' and l_t are turbulence intensity and integral scale, ν is the kinematic viscosity, $\tau_{l_t} = l_t/u'$ a large-eddy turnover time, $\tau_{\mathrm{ch}} = \delta_L/u_L$ a chemical (flame) time, and τ_η the Kolmogorov time at the dissipative scale.

These groups delineate the canonical regimes:

- Laminar / weak-turbulence: $\mathrm{Re}_t < 1$, the front is essentially laminar.
- Wrinkled/corrugated flamelets: $\mathrm{Ka} < 1$, $\mathrm{Da} < 1$, the inner structure remains laminar-like while turbulence primarily amplifies flame surface.
- Thickened–wrinkled flames: $1 < \mathrm{Ka} < 100$, $\mathrm{Da} < 1$; smaller eddies penetrate the preheat zone, effectively thickening the front and increasing stretch sensitivity.
- Broken reaction zones / well-stirred: $\mathrm{Ka} > 100$; reaction zones are strongly disrupted, and the process becomes mixing/chemistry controlled.

The turbulent premixed combustion regimes introduced above are reported in the Borghi diagram shown in Fig. 4.3.

In common 0D/1D engine modeling within the flamelet regime, chemistry is represented via engine-range correlations for $u_L(T, p, \phi, X_{\mathrm{dil}})$, while turbulence is introduced through (u', l_t) to control flame-surface amplification and the resulting burn rate [3]. Practically, the models are embedded in either a single-zone or a two-zone framework and differ mainly in how they represent surface amplification in the flamelet range (as organized in the Borghi diagram, Figure 4.3). The following subsections review the mainstream approaches with attention to assumptions, governing relations, inputs, and typical performance.

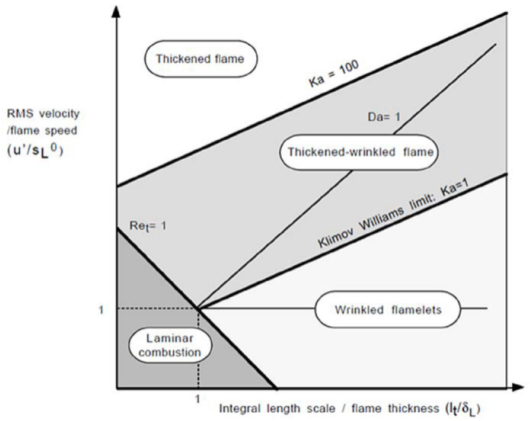


Fig. 4.3 Borghi (premixed turbulent combustion) diagram. Ordinate: normalized turbulence intensity u'/s_L^0; abscissa: normalized integral scale l_t/δ_L [5].

Single-zone. In the simplest formulation, the in-cylinder mixture is treated as spatially uniform. During closed-valve phases, the thermodynamic evolution can be advanced by writing the energy balance as

$$\frac{d(me)}{dt} = -p\frac{dV}{dt} - p\frac{dQ_w}{dt} \tag{4.6}$$

from which expressions for pressure and temperature follow. In particular, temperature evolves according to

$$\frac{dT}{dt} = \frac{1}{mc_v}\left(-p\frac{dV}{dt} - p\frac{dQ_w}{dt} - \frac{\partial e}{\partial x_b}\frac{dx_b}{dt}\right) \tag{4.7}$$

Change in volume is determined directly from engine geometry, while the wall heat-transfer term Q_w, which can be determined through the Woschni model [151], is usually correlated through dedicated submodels. The burned-mass fraction x_b is imposed to close the system; a common choice is the S-shaped Wiebe MFB law [152].

$$x_b(\theta) = 1 - \exp\left[-a\left(\frac{\theta-\theta_0}{\Delta\theta}\right)^{m+1}\right] \tag{4.8}$$

where $(a, m, \theta_0, \Delta\theta)$ are calibration parameters. This construction captures measured pressure traces robustly and with modest computational cost, though it remains non-phenomenological and typically requires case-specific tuning.

Two-zones. A more informative baseline divides the cylinder charge into unburned (u) and burned (b) zones that share a common pressure but retain distinct thermodynamic states; each zone satisfies its own energy balance while mass is transferred from u to b as combustion progresses.

$$\frac{d(m_b e_b)}{dt} = -p\frac{dV_b}{dt} - p\frac{dQ_{w,b}}{dt} + h_u\frac{dm_b}{dt} \tag{4.9}$$

$$\frac{d(m_u e_u)}{dt} = -p\frac{dV_u}{dt} - p\frac{dQ_{w,u}}{dt} - h_u\frac{dm_b}{dt} \tag{4.10}$$

The coupling term $h_u\frac{dm_b}{dt}$ accounts for the enthalpy carried with the mass that is converted from unburned to burned mixture; it links the two balances and makes the solution sensitive to the prescribed burn rate. Pressure is taken as spatially uniform in the cylinder, while temperatures, compositions, and transport properties differ between zones. Closing the model requires an estimate of the burn rate history $\frac{dm_b}{dt}$ (or equivalently the MFB profile), which is provided by a combustion submodel. Common choices range from empirical Wiebe-type laws, which are robust but non-phenomenological, to quasi-dimensional flame-propagation formulations that relate the instantaneous burning rate to the laminar flame speed u_L, turbulence intensity u', and an evolving flame surface constrained by chamber geometry. The latter embodies the classical dual role of chemistry (through u_L) and turbulence (through flame-front corrugation in the corrugated-flamelets regime), enabling realistic trends with speed, load, dilution, and ignition timing. The following sections review these closures and their implications for heat-release phasing, wall heat losses, and links to emissions and knock indicators.

Eddy burn-up. Within the two-zone framework outlined above, one of the most widely used closures is the eddy burn-up model, originating from Keck [153]. Combustion is idealized as two coupled stages consistent with the flame-brush framework discussed earlier: (i) turbulent entrainment of unburned mixture into a corrugated flame brush at a rate set by the available mean uncorrugated flame area A_L and an effective burning speed $u_L + u_T$, and (ii) exponential burn-up of the entrained mass into products over a characteristic time τ that reflects small-scale mixing and chemical conversion. In compact form,

$$\frac{dm_e}{dt} = \rho_u A_L (u_L + u_T) \tag{4.11}$$

$$\frac{dm_b}{dt} = \frac{m_e - m_b}{\tau} \tag{4.12}$$

$$\tau = \frac{\lambda}{u_L} \tag{4.13}$$

where u_L is the laminar burning velocity, $u_T \propto u'$ represents the turbulence-induced enhancement, m_e is the entrained, but not yet burned mass, and λ is a mixing length (often tied to an integral or brush thickness scale). Coupled to the two-zone energy balances at a single cylinder pressure, this combustion submodel staging supplies $\dot{m}_b$ without prescribing an a priori MFB curve, naturally yielding the observed S-shaped heat release and a realistic late-burn tail. Calibration is typically compact (entrainment and burn-up constants plus flame-area geometry factors) and targets CA10/50/90 and pressure traces across speed, load, and dilution, while known limitations include sensitivity to turbulence correlations and incomplete control of wall-phase slow-down.

Fractal model. Here [154] turbulence-induced surface amplification is described via a wrinkling factor A_T/A_L:

$$\begin{aligned} \frac{dm_b}{dt} &= \rho_u u_L A_T \\ &= \rho_u u_L \left(\frac{A_T}{A_L}\right) A_L \\ &= \rho_u u_L \left(\frac{l_{\max}}{l_{\min}}\right)^{D_3-2} A_L \end{aligned} \tag{4.14}$$

Within the two-zone framework, the burn rate is written as $\frac{m_b}{dt} = \rho_u u_L A_T$ with the turbulent flame area obtained from a fractal amplification of the mean (uncorrugated) area A_L. The fractal dimension D_3 increases with turbulence intensity; the outer cutoff $l_{\max}$ is linked to the integral scale or instantaneous flame size, while the inner cutoff $l_{\min}$ represents the smallest dynamically relevant scale, often correlated to the laminar flame thickness in the flamelet regime or to the Kolmogorov scale at higher Karlovitz. In this view, turbulence accelerates combustion primarily by wrinkling

the flame, and so increasing A_T, while the local burning speed remains u_L. The model integrates cleanly with the two-zones energy balances and supplies $\frac{m_b}{dt}$ without prescribing an a priori MFB curve; with a compact set of correlations for D_3, $l_{\max}$, and $l_{\min}$, it typically reproduces the S-shaped heat release, the burn rate peak, and trends with speed, load, and dilution. Compared head-to-head with eddy burn-up, fractal formulations often improve the late-burn tail and speed sensitivity, though performance remains sensitive to the chosen scale correlations and to assumptions of isotropic/homogeneous turbulence and geometry-dependent clipping.

Coherent Flame Model. The Coherent Flame Model (CFM) [155] belongs to the flame surface-density family and retains the same consumption form used in the two-zone framework,

$$\frac{m_b}{dt} = \rho_u u_L A_T = \rho_u u_L \Xi A_L \tag{4.15}$$

Here the wrinkling factor Ξ links the turbulent flame surface A_T to the mean (uncorrugated) surface A_L via $A_T = \Xi A_L$, so turbulence accelerates burning primarily by creating surface rather than by altering the local laminar speed u_L. Two ingredients make the model predictive at low cost and consistent with the two-zone balances: (i) the mean area A_L is spherical only for very small burns and is otherwise tabulated versus piston position and burned volume V_b to encode chamber geometry; (ii) Ξ evolves from a 0D ordinary differential equation obtained by reducing the 3D flame-surface-density transport [156], which supplies a dynamic balance between production by turbulent strain/curvature and destruction by propagation/dilatation and near-wall effects. A compact form is given by:

$$\frac{d\Xi}{dt} = \left(\frac{u'}{u_L}\Gamma\right)\frac{1}{l_t}\left(\Xi_{eq} - \Xi\right) - \left(\frac{\rho_u}{\rho_b} - 1\right)\frac{u_L}{r_b}\frac{\delta_L}{l_t} \tag{4.16}$$

$$\Xi_{eq} = 1 + C\frac{u'}{u_L}Sc^{-1/2} \tag{4.17}$$

where u' and l_t are turbulence intensity and integral scale, Γ is an efficiency function, $r_b = (3V_b/4\pi)^{1/3}$ is a mean flame radius, and δ_L is the laminar thickness. This formulation captures the laminar to turbulent transition, naturally reproduces the S-shaped heat release, and damps wrinkling under strong thermal expansion near

top dead center. In practice, initialization of Ξ, a small set of production/destruction and wall-interaction coefficients, and geometry-informed $A_L(V_b)$ tables suffice for calibration against CA10/50/90 and pressure traces across speed, load, and dilution sweeps. Compared with algebraic wrinkling laws (e.g., fractal formulations) and entrainment-based closures (e.g., eddy burn-up), CFM offers a dynamic, physically interpretable link between turbulence metrics and instantaneous burn rate without prescribing an a priori MFB curve, while remaining sensitive to the turbulence-scale correlations and the near-wall clipping strategy.

4.1.2 Data-driven

Extending the phenomenological framework introduced in Section 4.1.1, data-driven combustion models pursue the same overarching objective, such that, obtaining credible and predictive representations of the combustion process, but achieve it by learning the mapping from operating conditions and control inputs to combustion metrics directly from data. Depending on the intended application, the model output may consist of scalar cycle indicators (e.g., MFB50, burn duration, peak pressure) or a crank angle–resolved profile such as the burn rate or heat release rate. Rather than prescribing turbulence–chemistry interaction closures, the learned representation implicitly captures how these effects combine within the engine's operating envelope.

In many control and calibration contexts, scalar combustion descriptors are sufficient and often preferable. For embedded ECU implementation, static or low-order representations offer simplicity, robustness, and transparency in calibration. Under such constraints, reconstructing the full crank angle–resolved burn rate trace may introduce unnecessary computational and tuning complexity.

However, the availability of a complete burn rate profile becomes significantly more valuable in simulation-oriented environments. In real-time applications such as Hardware-in-the-Loop, virtual test rigs, and hybrid physics–data-driven digital twins, phase-resolved combustion modeling enables consistent coupling with thermodynamic and mechanical solvers. A full burn rate reconstruction preserves ignition delay, premixed and diffusion-controlled phases, and late-burn evolution, allowing multiple combustion indicators to be derived from a physically coherent representation. When integrated within co-simulation frameworks or physical engine test rigs, this detailed phase information ensures that pressure development, load-dependent

phasing effects, and late-burn dynamics propagate consistently through drivetrain and thermal subsystems. Such coherence is essential for assessing overall powertrain response and system-level interactions, even if the final production controller ultimately relies on simplified scalar models.

From a computational standpoint, phenomenological and CFD-based approaches provide strong physical interpretability but are often too demanding for large Design of Experiments (DoE) campaigns, iterative optimization loops, or real-time deployment [157]. Recent advances in ML and DL have mitigated these limitations by approximating nonlinear relationships between measurable engine variables—such as intake pressure, EGR rate, spark timing, and air–fuel ratio—and combustion descriptors directly from experimental data [158, 159]. By bypassing explicit thermochemical solution steps, neural networks achieve high predictive capability at millisecond-level inference times.

A wide range of architectures has been explored in the literature. Deep feedforward networks have demonstrated highly accurate in-cylinder pressure prediction for HCCI operation [158]. Recurrent architectures such as LSTMs and hybrid CNN–LSTM models have successfully captured pressure dynamics across broad operating ranges, including variations in intake temperature and EGR dilution [160, 161]. Beyond pressure reconstruction, ML-based predictors have targeted combustion phasing and emissions metrics: [159] reported accurate predictions of CA50, $\mathrm{HRR}_{\mathrm{max}}$, and burn duration in biodiesel-fueled engines, while [28] combined Gaussian processes with ANNs to reconstruct the mass fraction burned (MFB) trajectory with near-phenomenological accuracy at substantially reduced computational cost. Hybrid physics–ML formulations further demonstrate that thermodynamic consistency can be preserved through constrained loss functions while retaining real-time capability [157]. These developments support applications in digital twins, virtual sensing, and closed-loop simulation [27]. Recurrent models such as LSTMs and GRUs have additionally shown strong capability in capturing dilution and combustion phasing effects relevant to low-temperature and hydrogen-fueled operation [162].

The present modeling framework is particularly suited to highly diluted spark ignition regimes, including lean-burn strategies, high-EGR operation, catalyst heating phases with retarded combustion phasing, and advanced dilution concepts. In these operating domains, combustion stability, late-burn behavior, and phasing sensitivity

are strongly influenced by dilution level, making phase-resolved prediction especially beneficial.

Despite the rapid evolution of data-driven combustion modeling, several limitations remain evident in the literature. First, many studies prioritize scalar descriptors over full crank angle–resolved burn-rate reconstruction. Second, high-dilution spark-ignition regimes, where late-burn dynamics and stability constraints are most critical, remain comparatively underexplored. Third, some approaches either constrain learning tightly to predefined phenomenological forms or directly regress the in-cylinder pressure trace. While pressure regression enables subsequent derivation of net heat release via thermodynamic analysis, the combustion-phase evolution remains implicit within the learned mapping. In contrast, explicitly reconstructing the burn-rate profile under weak phenomenological constraints preserves combustion phase structure, enhances interpretability, and facilitates coupling across different thermodynamic backbones.

To address these gaps, the present work proposes a sequence-based LSTM trained on experimental data from a highly diluted spark-ignition engine to directly predict the crank angle–resolved burn-rate evolution over an entire cycle. Using inputs such as engine speed, load, dilution level, and spark timing, the model outputs the complete burn-rate trace without explicit turbulence–kinetics closures or manually tuned entrainment parameters. When coupled with standard single- and two-zone thermodynamic formulations, the approach would be able to deliver real-time pressure and heat-release histories with accuracy comparable to calibrated phenomenological baselines within the trained operating envelope. In doing so, it bridges the gap between physically detailed but computationally intensive phenomenology and fast, generalizable data-driven simulation frameworks suitable for real-time virtual testing.

It should be noted that the validation presented in this work focuses primarily on steady-state operating conditions. While the sequence-based LSTM architecture inherently captures cycle-wise dependencies and is expected to improve robustness under varying operating conditions compared to static regressors, no dedicated validation under strongly transient maneuvers is included. Accordingly, no explicit claim of superiority in highly transient operation is made. Extending validation toward aggressive transients represents a natural continuation of this research.

4.2 Case Study

The learning-based combustion framework was applied to experimental data from the European Horizon 2020 PHOENICE project. The reference engine is a state-of-the-art four-cylinder, 1.3 L, turbocharged, Gasoline Direct Injection (GDI) SI engine [163]. It features a high compression ratio, a compact four-valve combustion chamber with a 200 bar fuel injection system, and a MultiAir Variable Valve Actuation (VVA) mechanism acting on the intake valves [164]. The cylinder head incorporates an integrated exhaust manifold, improving thermal efficiency and reducing packaging constraints. To achieve a target Indicated Thermal Efficiency (ITE) of 47%, the Dual Dilution Combustion Approach (DDCA) [165] was implemented. This strategy combines homogeneous lean combustion with cooled low-pressure EGR and operates under a high compression ratio. Several advanced subsystems were introduced to enable DDCA operation, particularly focusing on the optimization of the combustion system.

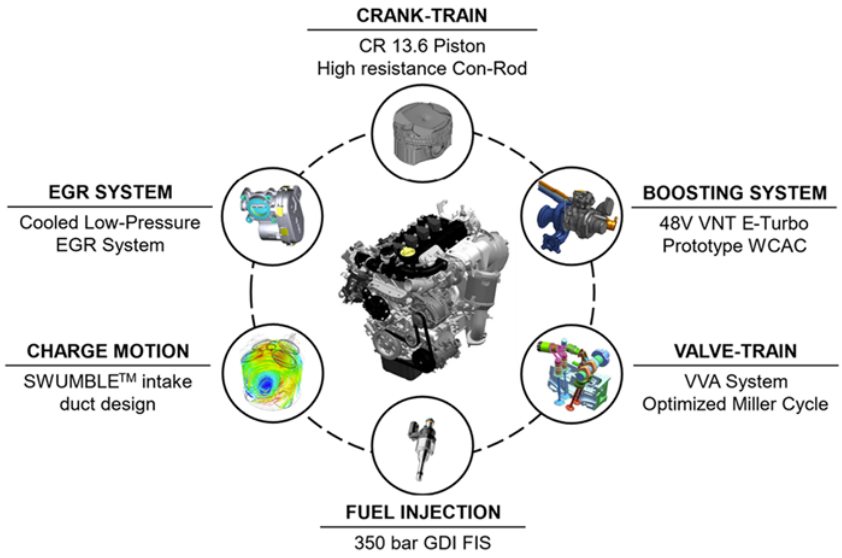


Fig. 4.4 Main technological upgrades implemented on the PHOENICE engine.

As shown in Figure 4.4, a major modification concerns the piston geometry, which increased the compression ratio from 10.5:1 to 13.6:1. Combined with a redesigned intake port, this configuration enabled the exploitation of the Swumble concept, enhancing turbulence intensity and flame propagation under lean and highly diluted conditions.

The boosting and fuel injection systems were also upgraded. A 48 V electrically assisted turbocharger with a Variable Nozzle Turbine (VNT) was introduced to mitigate turbo lag and allow energy recovery when turbine power exceeds compressor demand. In parallel, a high-pressure injection system capable of operating up to 350 bar was implemented, improving atomization and mixture preparation. Finally, the VVA system was exploited to enable aggressive Miller-cycle operation, reducing pumping losses and improving efficiency. The final engine configuration and specifications are summarized in Table 4.1.

Table 4.1 Main specifications of the PHOENICE engine.

Engine Specifications	
Number of cylinders	4
Displacement	1332 cm^3
Bore $\times$ Stroke	70 mm $\times$ 86.5 mm
Compression Ratio	13.6:1
Number of valves	16
VVA system	MultiAir III (intake only)
Turbocharging	48 V VNT E-Turbo
Fuel Injection	GDI (up to 350 bar)
Ignition System	Base production
EGR System	Cooled Low Pressure (LP)
Rated Power (target)	100 kW @ 4500 RPM
Rated Torque (target)	218 Nm @ 3500 RPM

Experimental Dataset

The dataset used in this study was obtained during the calibration phase of the PHOENICE engine, conducted by IFP Energies Nouvelles (IFPEN). The campaign consisted of steady-state measurements collected at eleven representative operating points, selected to span a wide range of engine speeds and loads, as shown in Figure 4.5 and summarized in Table 4.2.

At each point, the air–fuel ratio (λ) and EGR rate were systematically varied to quantify the combined effect of dilution and lean operation. In total, 101 distinct operating conditions were acquired, each corresponding to a specific combination of engine speed, load, λ, and EGR rate. Two representative examples of the test matrices, at 1500 RPM $\times$ 5.5 bar BMEP and 3000 RPM $\times$ 7 bar BMEP, are reported in Tables 4.3 and 4.4.

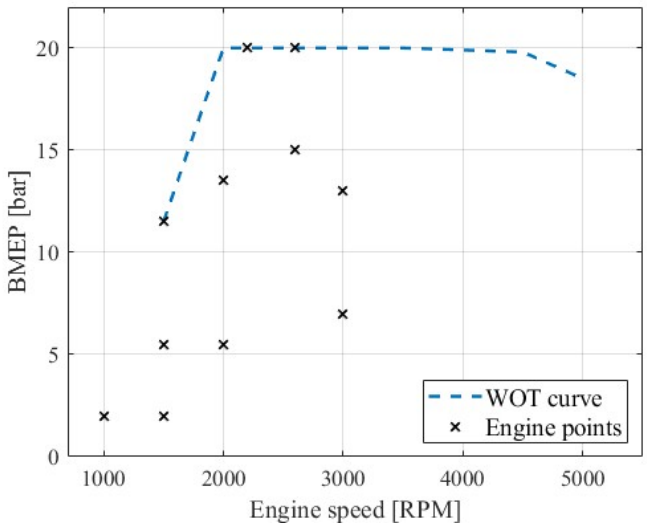


Fig. 4.5 Selected engine operating points and full-load curve.

For each operating condition, IFPEN recorded a comprehensive set of variables, including air and fuel mass flow rates, pressures and temperatures at the compressor and turbine inlets and outlets, BSFC. Additional parameters such as turbocharger speed and Spark Advance (SA) were also acquired. Finally, the burn rate curves employed in this work were derived from a GT-Suite simulation model previously calibrated using experimental data from IFPEN. These curves serve as the reference for analyzing combustion characteristics and validating the learning-based predictive framework developed in this study. It is worth emphasizing, however, that GT-Suite is used here primarily as a post-processing tool to extract consistent burn rate profiles from the experimentally measured in-cylinder pressure traces, rather than as a predictive surrogate to generate combustion data for unseen operating conditions. Therefore, the proposed learning-based model is effectively trained and assessed against combustion information that remains rooted in experimental measurements, while acknowledging that reliance on a calibrated simulation tool for burn rate reconstruction may still introduce modelling errors, particularly under highly diluted or extreme transient conditions.

Table 4.2 Tested steady-state engine points.

Engine Speed [RPM]	BMEP [bar]
1000	2
1500	2
1500	5.5
1500	11.5
2000	5.5
2000	13.5
2200	20
2600	15
2600	20
3000	7
3000	13

Table 4.3 Air–fuel ratio and EGR sweeps at 1500 RPM $\times$ 5.5 bar BMEP.

Air–Fuel Ratio [–]	EGR Rate [%]
1.00	0.9, 5.2, 10.2, 15.3, 18.9
1.11	0.7, 10.3, 15.0
1.25	0.6, 5.1
1.43	0.0

4.3 Methodology

This section outlines the pipeline adopted to obtain real-time, crank–angle–resolved combustion models. The workflow proceeds in four steps. First, a compact and physically plausible input set is established through a two-stage feature selection process: a statistical screen (correlation analysis and Neighborhood Component Analysis) followed by a physics check to retain variables with causal influence on phasing and duration. Second, empirical feature distributions are analyzed to assess coverage, identify sparsely sampled regimes, and anticipate extrapolation risk. Third, two complementary models are constructed: (i) a fast NN–Wiebe model that infers single-Wiebe coefficients from the selected features after per-point calibration via a Genetic Algorithm; and (ii) a hybrid recurrent model (Hybrid–LSTM) that predicts the full MFB sequence while relaxing the S-shaped prior. The sequence model employs a dual-branch architecture (static feature encoder and LSTM encoder) and a physics-informed objective that enforces pre-ignition behavior and derivative consistency with the target burn rate.

Table 4.4 Air–fuel ratio and EGR sweeps at 3000 RPM × 13 bar BMEP.

Air–Fuel Ratio [–]	EGR Rate [%]
1.00	0.0, 5.2, 10.0, 15.2, 20.6, 21.5
1.11	0.0, 5.3, 9.8, 15.1, 20.1
1.25	0.0, 5.1, 10.0, 14.4
1.43	0.0, 5.2, 7.2

4.3.1 Feature Selection

This subsection consolidates a compact and physically meaningful input set for the Wiebe–NN and Hybrid–LSTM models. The starting pool comprised standard actuations and states from the PHOENICE dataset (engine speed, load/IMEP or BMEP, air–fuel ratio λ, EGR rate, spark timing, and air-path variables), listed in Table 4.5. The selection strategy combines a statistical screen, used to remove redundancy and highlight task relevance, with a physics check to ensure that the retained variables carry plausible causal influence on combustion phasing and duration.

Table 4.5 Initial set of features after first human screening.

Initial Features	
Engine speed	[RPM]
Turbo speed	[RPM]
EGR	[%]
Air–fuel ratio λ	[-]
Intake pressure	[bar]
Intake temperature	[K]
Intake valve opening (IVO)	[deg CA]
Intake valve closure (IVC)	[deg CA]
Injected fuel mass	[mg/cycle]
Injection duration	[deg CA]
Start of injection (SOI)	[deg CA]
Spark advance (SA)	[deg CA]

First, linear dependence among candidates was quantified with Pearson's correlation coefficient,

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{4.18}$$

computed on the entire dataset. The numerator measures sample covariance, the denominator rescales by the product of standard deviations, and $r \in [-1,1]$ is invariant to units. As a rule of thumb, $|r| < 0.3$ indicates weak association, $0.3 \leq |r| < 0.7$ moderate, and $|r| \geq 0.7$ strong; the sign gives direction. The correlation map in Figure 4.6 shows tight blocks among thermodynamic channels and derived quantities; therefore, highly collinear pairs were collapsed to a single representative, favoring variables with clearer physical interpretation for combustion modeling.

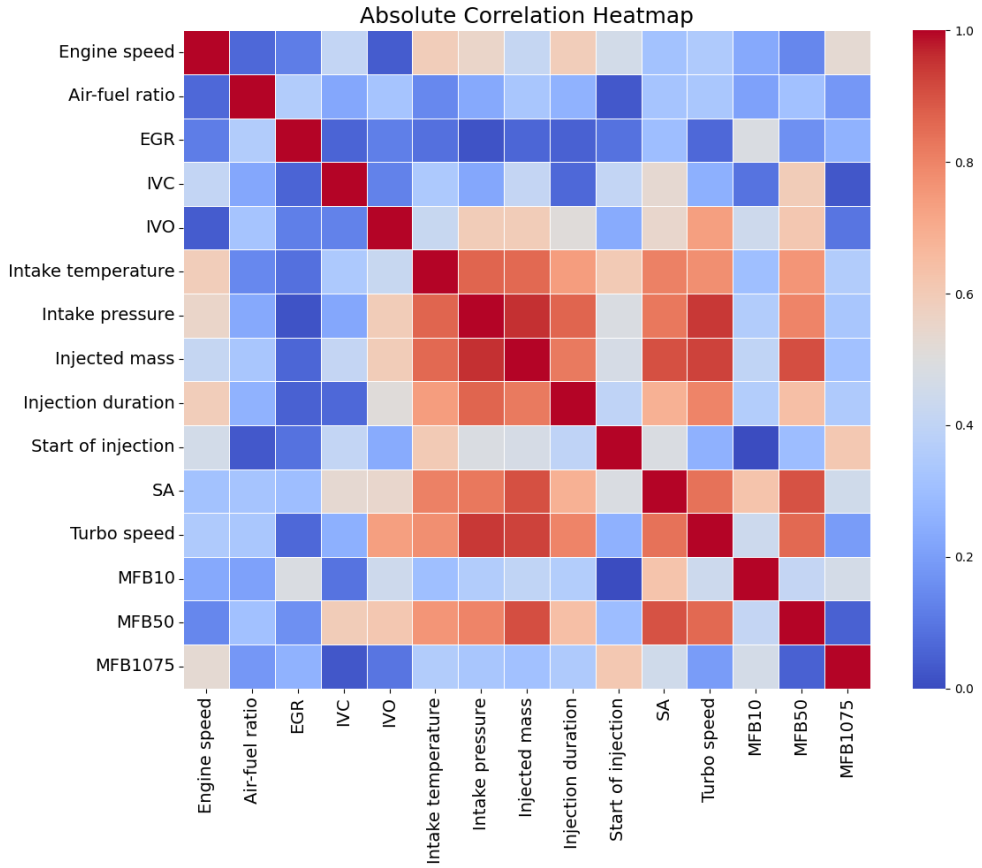


Fig. 4.6 Correlation matrix among candidate input features. Blocks of high correlation motivate redundancy pruning prior to supervised modeling.

Second, task relevance was quantified with Neighborhood Component Analysis (NCA) in regression form, which learns a data-driven scaling of inputs that maximizes local agreement between features and targets; the resulting weights provide an importance ranking.

Neighborhood Component Analysis results. To give metric-specific guidance, NCA was run separately for MFB10 for the early phasing, MFB50 for the center of combustion, and MFB10–75 for the duration. The profiles in Figure 4.7 indicate complementary roles: for MFB10, SA dominates with EGR next, consistent with ignition phasing and dilution governing onset; for MFB50, injected fuel mass leads, followed by SA and boost-related variables; for MFB10–75, SOI and engine speed are most influential, reflecting mixture-preparation effects and time-to-crank-angle conversion. Taken together, the statistical screen and the NCA ranking point to a compact set tailored to the study. Specifically, IVO/IVC were removed due to low correlation with the targets under the Miller-oriented phasing, injection duration was dropped as redundant with injected mass, and SOI showed limited relevance for the combustion profiles considered. The eight variables retained for all subsequent analyses are summarized in Table 4.6.

Table 4.6 Final set of features used across the analysis.

Final Features	
Engine speed	[RPM]
Turbo speed	[RPM]
EGR	[%]
Air-fuel ratio λ	[-]
Intake pressure	[bar]
Intake temperature	[K]
Injected fuel mass	[mg/cycle]
Spark advance	[deg CA]

It should be noted that this pipeline is purely statistical and does not encode prior physical knowledge; rankings can therefore be affected by correlations or non-causal associations. Consequently, the final subset was validated against combustion physics to retain only features whose statistical relevance is consistent with plausible mechanisms.

4.3.2 Feature Distribution Analysis

The empirical distributions of the selected variables provide the context that explains the rankings above and indicate where model generalization is most credible. Co-variations are consistent with engine physics: injected fuel mass and intake pressure shift to higher values at larger loads; SA moves with dilution; valve events remain

narrowly distributed under the Miller strategy. These patterns confirm that the core set in Section 4.3.1 sits on well-covered regions of the feature space, while also flagging sparse regimes—particularly extreme dilution and high load—where prediction is more challenging and additional testing would most effectively enrich the dataset.

The empirical distributions of the selected variables provide the context that explains the rankings above and indicate where model generalization is most credible. The PDFs reproduce an experimental campaign built around a discrete speed–load matrix with systematic dilution sweeps, since several inputs display clustered or step-like support, like speed and BMEP at scheduled points, $\lambda \in \{1.00, 1.11, 1.25, 1.43\}$, and EGR spanning 0–22%. These observations were used to check that the statistical selection in Section 4.3.1 aligns with physically plausible coverage and to identify regions with limited support for later discussion. The key outcome of the distribution analysis can be summarized in the following points:

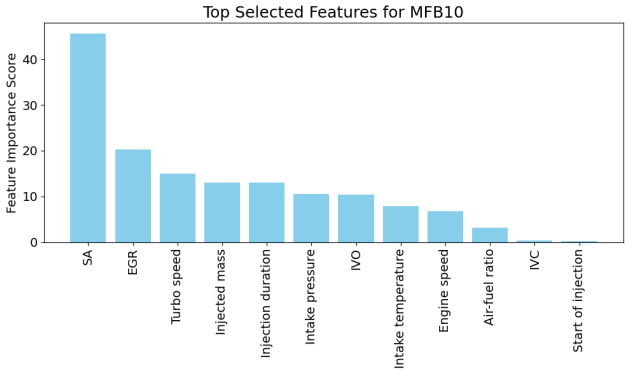
- **Engine speed & Load (BMEP).** The distributions are multimodal, mirroring the discrete speed–load grid of the campaign. Speed shows two main modes (low–mid and mid–high ranges), while BMEP concentrates at low/mid loads with a secondary lobe at mid/high loads and a thin tail at the highest loads. This indicates strong coverage of the central operating envelope and sparser sampling at the extremes, where knocking limits the engine operation in those regions.

- **Dilution: EGR & Air–fuel ratio λ.** EGR density decreases roughly monotonically with a long right tail up to 20%, showing emphasis on low–to–moderate dilution and fewer heavily diluted cases. The λ PDF is step-like/right-skewed with shoulders at the scheduled setpoints ($\lambda \approx 1.00, 1.11, 1.25, 1.43$), consistent with planned lean/stoichiometric.

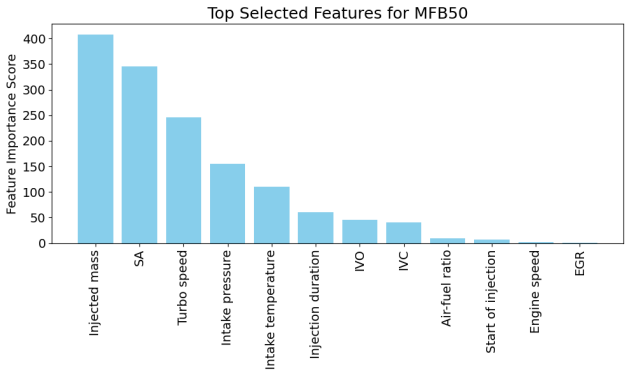
- **Air path: Turbo speed, Intake pressure, and Intake temperature.** Turbo speed is strongly left-skewed with a bimodal shape, with most mass at low shaft speeds and a light high-speed tail; p_{in} shows a broad, slightly bimodal structure aligned with the two BMEP bands; T_{in} varies within a compact range and exhibits mild bimodality, reflecting a distribution in line with the intake pressure.

- **Spark advance (SA).** The distribution is concentrated at negative crank angles, with a primary mode at moderate advance and a secondary shoulder at smaller advance. This pattern is consistent with retard under high load or high dilution for knock/stability management and with the scheduled dilution sweeps.

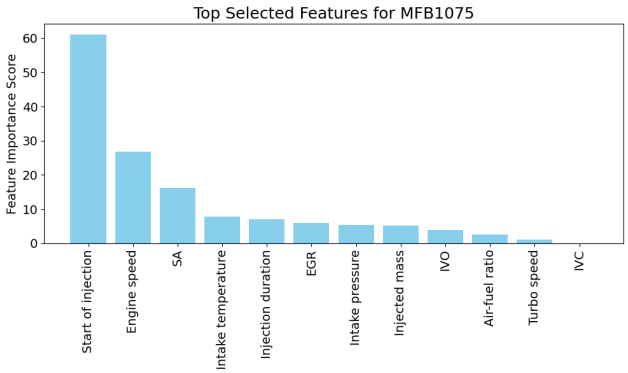
This distributional analysis highlights likely failure regions for the models—operating conditions that are sparsely represented in the dataset. In such areas, the networks must extrapolate rather than interpolate, which typically reduces accuracy. Late/delayed-combustion regimes are particularly underrepresented and thus more error-prone, as shown in Section 4.4. The analysis also points to where additional engine testing should be prioritized—targeting rare speed–load–dilution combinations—to enrich the dataset, broaden coverage, and improve model robustness.



(a)

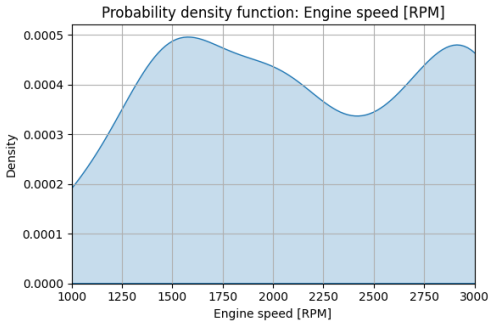


(b)

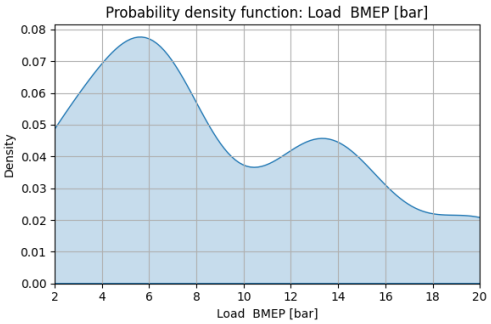


(c)

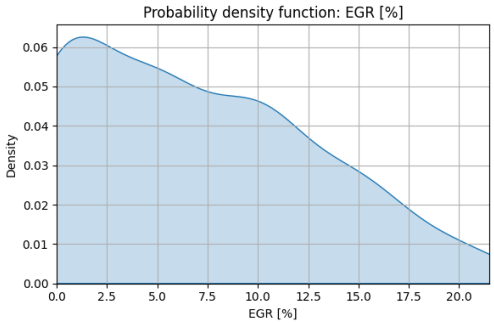
Fig. 4.7 NCA feature importance profiles: (a) MFB10, (b) MFB50, (c) MFB10–75.



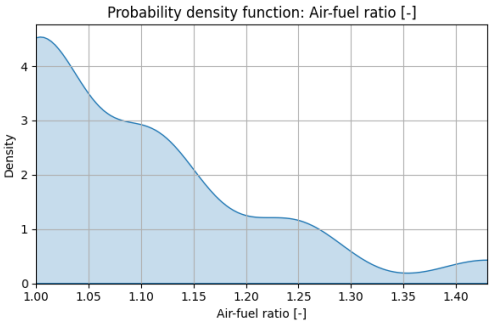
(a) Engine speed



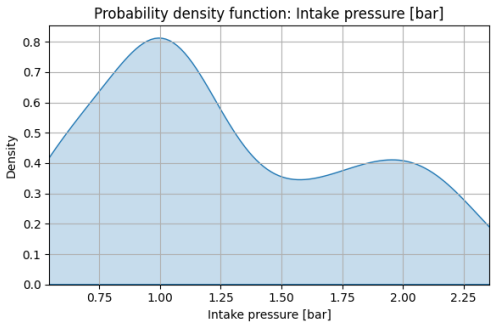
(b) Load (BMEP)



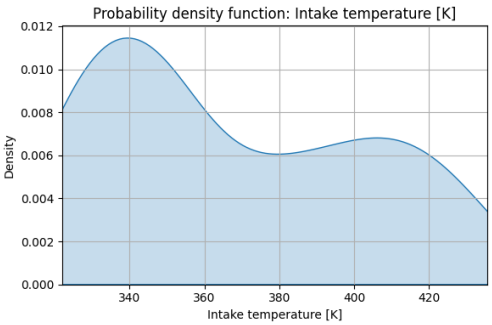
(c) Exhaust Gas Recirculation (EGR)



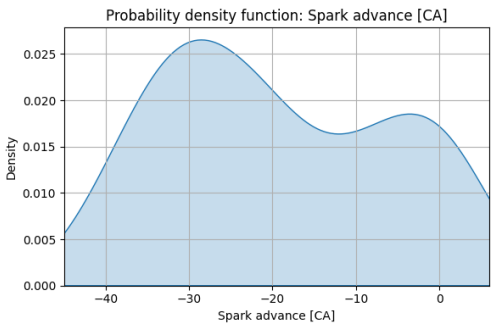
(d) Air–fuel ratio λ



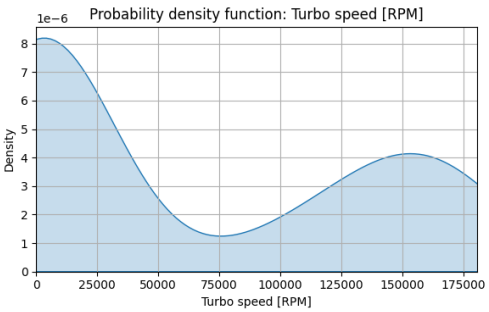
(e) Intake pressure



(f) Intake temperature



(g) Spark advance



(h) Turbo speed

Fig. 4.8 Probability density functions of selected features.

4.3.3 Neural Network for Wiebe Calibration

Having established a compact, physics-plausible input set (Table 4.6) and clarified its coverage through the distribution analysis, the next step is to deploy a fast and interpretable surrogate that maps those features to a full combustion trajectory. To balance robustness in sparsely sampled regimes with fidelity over the well-covered envelope, a compact feed-forward NN is coupled to the single-Wiebe analytical form: the NN predicts the four Wiebe coefficients from the selected features (speed, load, λ, EGR, SA, and air-path states), while the analytic law reconstructs the MFB/burn rate at negligible cost. The workflow is two–stage: first, for each operating point a reference parameter set is obtained by fitting the Wiebe curve to the target combustion trace via a Genetic Algorithm (GA) [147]; second, a supervised network learns the mapping from features to those GA-optimized parameters. This design leverages the physical structure highlighted by the feature selection (e.g., the dominant role of SA and dilution on phasing) and provides stable, real-time predictions aligned with the dataset's coverage.

Wiebe form and fitting objective. Within a crank-angle window $\theta \in [\theta_{\min}, \theta_{\max}]$ surrounding combustion, the cumulative burned-mass fraction $x_b(\theta)$ is modeled with the single Wiebe law in Equation (4.8); the burn rate profile follows by differentiation, $\dot{x}_b(\theta) = \frac{dx_b}{d\theta}$. The calibration parameters $(a, \theta_0, \Delta\theta, m)$ at each operating point are identified by minimizing the discrepancy between the analytic curve and the target trace sampled on a discrete crank-angle grid $\{\theta_i\}_{i=1}^{N_\theta}$. Throughout, the MFB signal is fitted; the corresponding least-squares cost is

$$J_{\mathrm{GA}} = \sum_{i=1}^{N_\theta} \Big[x_b(\theta_i) - \hat{x}_b(\theta_i) \Big]^2, \tag{4.19}$$

with $\hat{x}_b$ the Wiebe reconstruction. The GA settings adopted are summarized in Table 4.7; the configuration favors broad exploration (large population, uniform crossover, relatively aggressive mutation) with a saturation-based stop to avoid over-tuning. Figure 4.9 illustrates a typical fit: the S-shaped evolution and peak region are captured well, while small deviations can appear at the very onset and tail—limitations intrinsic to the single-curve structure and a motivation for the hybrid sequence model introduced later.

Table 4.7 Genetic Algorithm configuration used for Wiebe parameter optimization.

Genetic Algorithm Configuration	
Number of generations	10000
Parents per generation	4000
Population size	9000
Number of genes	4
Mutation	50% of genes, random
Crossover	Uniform
Stop criterion	Saturation over 50 generations

Learning the parameter map. To enable online prediction, a small multilayer perceptron is trained on pairs $\left(z_i, \hat{\psi}_i\right)$, where z_i collects standardized inputs (zero mean, unit variance per feature) and $\hat{\psi}_i = [\hat{a}, \hat{m}, \hat{\theta}_0, \Delta\hat{\theta}]$ are the GA-optimized targets. Following a modest hyperparameter grid search (depth/width, activation, regularization), the selected architecture uses *three hidden layers with 64-32-16 neurons* and a tanh activation with a 4-neuron linear output layer, which provided the best validation error and a smooth mapping across operating conditions. The training loss is the mean-squared error on parameters,

$$\begin{aligned} J_{\text{NN–Wiebe}} &= \sum_{i=1}^{N} \| \hat{\psi}_i - \psi_i \|_2^2, \\ \psi_i &= [a, m, \theta_0, \Delta\theta], \\ \hat{\psi}_i &= [\hat{a}, \hat{m}, \hat{\theta}_0, \Delta\hat{\theta}] \end{aligned} \tag{4.20}$$

Early stopping is used to promote smoothness and prevent overfitting. At inference, the network outputs ψ in a single pass; these coefficients are inserted into Equation (4.8) (or its derivative) to reconstruct the MFB (or burn rate), from which CA10/50/90 and duration are computed analytically.

4.3.4 Hybrid Recurrent Neural Network Model for Burn Rate Prediction

While the NN–Wiebe surrogate in Section 4.3.3 offers a compact and interpretable mapping from operating conditions to a full burn rate profile, its accuracy is ultimately bounded by the expressiveness of the single-Wiebe function. In particular, the intrinsic S-shaped constraint embedded in the Wiebe law can lack in accurately

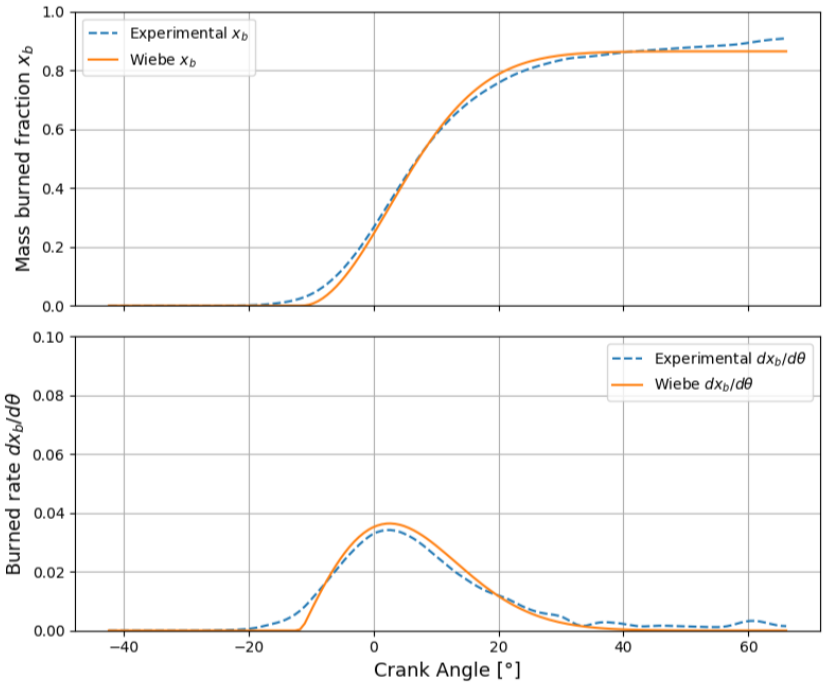


Fig. 4.9 Example GA fit of the Wiebe MFB curve against the reference combustion trace. The central burn is matched closely; minor discrepancies at onset and late tail reflect the expressiveness of a single-Wiebe law.

describing combustion regimes that deviate from premixed-like behavior. This mismatch is especially pronounced for non-premixed (diffusive) combustion, for cases with strong dilution (high EGR or very lean operation), and for cycles with a retarded start of combustion, where the mass fraction burned trajectory deviates from a canonical sigmoid evolution.

The case study engine employed in this work employs several of these features by a design choice aimed at pushing the engine efficiency through high dilution and phasing strategies (see Section 4.2). Consequently, it can benefit from a combustion description that is not constrained to an S-shape mathematical description. As illustrated in Figure 4.9, the single-Wiebe fit reproduces the combustion phasing well, but tends to underperform at the start of the combustion process and on its tails. These start and tail discrepancies are consistent with the limited flexibility of an exponential-like cumulative law and motivate relaxing the parametric constraints.

A possible extension, commonly adopted in Diesel combustion modeling, would be the use of a double Wiebe formulation, in which separate Wiebe functions are

employed to represent premixed and diffusion-controlled heat-release phases. While such an approach could improve the fit quality at both the beginning and the end of combustion, it also introduces additional degrees of freedom that are primarily mathematical in nature. Unlike the single Wiebe formulation, whose parameters are directly linked to physically meaningful quantities such as combustion phasing, start of combustion, and burn duration, a double-Wiebe model weakens this direct physical interpretability, as the decomposition into multiple Wiebe components is not uniquely linked to the underlying engine operating conditions. In a data-driven context, this loss of physical anchoring may be detrimental: the increased parameterization can obscure the relationship between inputs (operating conditions) and outputs (combustion evolution), potentially leading to poor generalization or unstable training when embedded within an NN framework. For this reason, simply increasing the complexity of the Wiebe formulation does not necessarily address the core modeling challenge.

To overcome these limitations within a data-driven framework, the search for a suitable functional form should not be restricted to the Wiebe family. Instead, the model should have greater freedom to find the most appropriate shape, allowing it to capture asymmetric rise, extended tails, and other non-sigmoidal patterns observed under different combustion regimes. This motivated the adoption of a purely NN-described burn rate evolution that uses a sequence modeling approach while preserving the strengths of feature-informed prediction and removing the hard S-shaped prior, thereby improving fidelity at the early and late stages of combustion without sacrificing the accurate prediction of the combustion phasing, which is very well captured by the Wiebe-imposed model. Designing an effective ML architecture requires aligning the model structure with the data characteristics and with the output objective. As highlighted in Figure 4.10, the ML model elaborates on two distinct data types: (i) a crank angle–resolved burn rate trajectory that is inherently sequential, and (ii) static operating conditions (e.g., load, speed, EGR, equivalence ratio, injection timing, boost) that remain constant over an engine cycle. Treating these sources symmetrically would either dilute the temporal inductive bias or force constant features into a recurrent pipeline where they offer no sequence information and risk biasing the output prediction. Accordingly, the model adopts a dual-branch design that processes sequential and constant features in parallel and combines them only after the respective embeddings have been formed.

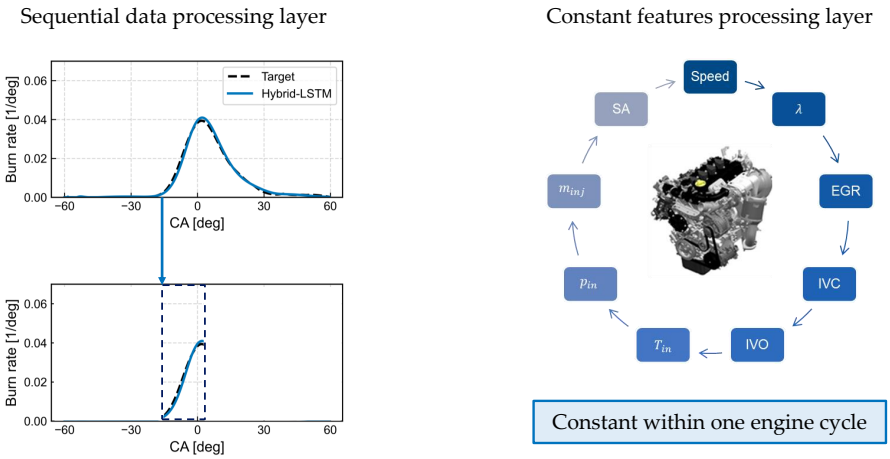


Fig. 4.10 Features data type for burn rate prediction: two parallel branches process the sequential burn rate and crank angle trends and the static, constant-feature inputs, respectively.

For the sequential branch, an LSTM is employed (see Section 2.1.2) to capture temporal dependencies across crank angle. This choice is motivated by three considerations: (i) the burn rate evolution exhibits path-dependent structure (e.g., ignition delay, rapid heat release rise, tailing) that benefits from gated memory and controlled state updates; (ii) the LSTM's hidden and cell states compactly summarize output trajectory history, reducing the need for explicit autoregression; and (iii) the burn rate output is not fed back as an input, thereby avoiding error accumulation across crank angle while retaining the dynamics in the internal states. In parallel, the static branch maps the constant operating conditions through a compact fully connected network to produce a context embedding. Two practical design choices are used. First, static features are standardized, with a z-score method, reported in Equation (4.21), with x_i as the original signal, and μ and σ its mean and standard deviation, respectively.

$$z_i = \frac{x_i - \mu}{\sigma} \tag{4.21}$$

Second, the static branch produces an operating conditions context vector, which is then concatenated with the LSTM sequence embedding at each crank angle step, enabling step-wise predictions to be conditioned on a consistent description of the operating point without reprocessing static features as if they were time-varying. A final fully connected layer maps the concatenated representation to the mass

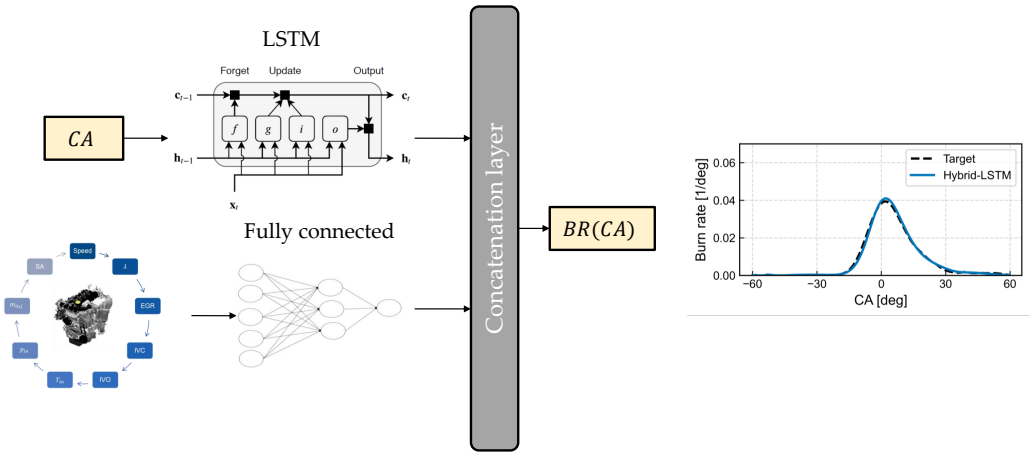


Fig. 4.11 Hybrid Recurrent model for burn rate prediction architecture.

fraction burned (MFB). The burn rate profile is then obtained by differentiating the MFB signal with respect to crank angle. Predicting the cumulative quantity (MFB) stabilizes the training and simplifies the functional search space for the network, while also enabling a derivative-based regularization term in the loss, i.e., a constraint proportional to the inferred burn rate, to promote physically plausible rise and tail behavior. A schematic representation of the mode architecture is reported in Figure 4.11.

The final model architecture and hyperparameters are reported in Table 4.8

Table 4.8 Hyperparameters of the hybrid-LSTM model.

Hyperparameter	**Value**
LSTM encoder nodes	256
Constant encoder nodes	256
Decoder nodes	512
Training algorithm	Adam
Learning rate	0.001
Epochs	1000
Mini-batch size	32

Each engine cycle is aligned to a combustion window spanning -60CAD before top dead center (BTDC) to $+60$CAD after top dead center (ATDC), specializing the model on the main combustion phase and avoiding non-informative portions of the cycle. Sequences are not divided into subsequences; the LSTM operates on full

cycles so that its gated memory can determine which parts of the past trajectory are salient for the current prediction under the given operating conditions.

Training adopts a physics-informed strategy that injects domain knowledge, imposes soft constraints, and reduces reliance on purely data-driven curve fitting. The overall objective in Equation (4.22) averages a per-step loss batch elements and cumulates across the entire crank angle sequence. The per-step loss in Equation (4.23) consists of: (i) a data-fidelity term enforcing agreement between predicted and target MFB, (ii) a pre-ignition constraint active for $\theta_t < \theta_{\mathrm{SA}}$ that suppresses non-zero MFB before spark timing, and (iii) a derivative-regularization term that aligns the derivative of the predicted MFB with the target burn rate profile. Lastly, the loss is accumulated over the entire cycle and averaged across the batch, as formalized in Equation (4.22); two weighting coefficients scale the physics terms to comparable magnitude with the data term, supporting generalization to unseen operating points.

$$J_{\text{Hyb-LSTM}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{\Theta}\ell_{t,i}, \tag{4.22}$$

with the per-step loss defined as

$$\ell_{t,i} = \begin{cases} \big(1+\lambda_{\text{pre}}\big)\big(\hat{x}_{b,i}(\theta_t) - x_{b,i}(\theta_t)\big)^2 + \lambda_{\text{deriv}}\left(\left.\dfrac{d\hat{x}_{b,i}}{d\theta}\right|_{\theta_t} - \left.\dfrac{dx_{b,i}}{d\theta}\right|_{\theta_t}\right)^2, & \theta_t < \theta_{\mathrm{SA}}, \\ \big(\hat{x}_{b,i}(\theta_t) - x_{b,i}(\theta_t)\big)^2 + \lambda_{\text{deriv}}\left(\left.\dfrac{d\hat{x}_{b,i}}{d\theta}\right|_{\theta_t} - \left.\dfrac{dx_{b,i}}{d\theta}\right|_{\theta_t}\right)^2, & \theta_t \geq \theta_{\mathrm{SA}}. \end{cases} \tag{4.23}$$

4.4 Results

The objective of this section is to evaluate data-driven combustion models that (i) accurately reconstruct the burn rate profile (peak intensity, phasing, and duration), (ii) generalize consistently across training, validation, and test splits, and (iii) respond correctly to changes in operating conditions (speed, load, EGR, and λ). In other words, beyond achieving low error on held-out data, the models are required to preserve the physical evolution of combustion as the operating point varies.

Two complementary approaches are examined. The first is a feed-forward NN that maps operating conditions to the coefficients of a single-Wiebe law calibrated via a GA, after which the burn rate is reconstructed from the inferred parameters (Section 4.3.3). The second is a hybrid recurrent architecture (Hybrid-LSTM) that predicts the burn rate sequence directly, blending domain structure with sequence learning to capture temporal features of the combustion process (Section 4.3.4).

The evaluation proceeds in three steps. First, we assess learning stability and potential over/underfitting by analyzing the training and validation losses and by inspecting representative reconstructions (Section 4.4.1). Second, we probe physical consistency through operating-condition sweeps—speed/load and dilution (EGR–λ), to verify that the predicted burn rate profiles evolve plausibly with the underlying physics and that performance is uniform across data splits. Third, we quantify generalization over the full dataset using standard combustion metrics (MFB10, MFB50, and MFB10–75), reported as regression plots and error statistics. Throughout, comparisons with the Wiebe-parameter model are provided to highlight the benefits and remaining limitations of each approach.

4.4.1 Hybrid-LSTM Training Assessment

The Hybrid-LSTM exhibits stable and well-aligned learning dynamics. As shown in Figure 4.12, both training and validation losses decrease smoothly during the initial epochs and converge to nearly identical stationary values. The absence of oscillations and the small gap between the curves indicate reliable convergence without signs of undertraining. The near overlap of the terminal losses suggests no appreciable overfitting and is consistent with good generalization to unseen cycles.

These learning dynamics translate into accurate burn rate reconstructions. In Figure 4.13a, a representative training cycle is reproduced with high fidelity, including both the rising and decaying flanks of the heat-release peak. Comparable accuracy is retained on validation data (Figure 4.13b), even for a case with markedly delayed combustion, an operating condition that is comparatively rare in the dataset. Despite this scarcity, the prediction closely follows the target in both magnitude and shape, indicating that the Hybrid-LSTM has learned robust temporal features rather than merely memorizing frequent patterns. Overall, the agreement between losses and the quality of the reconstructions support the conclusion that the model

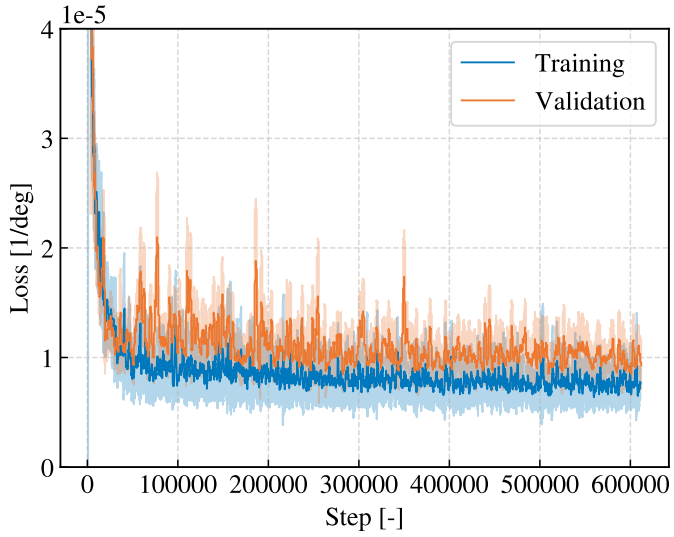
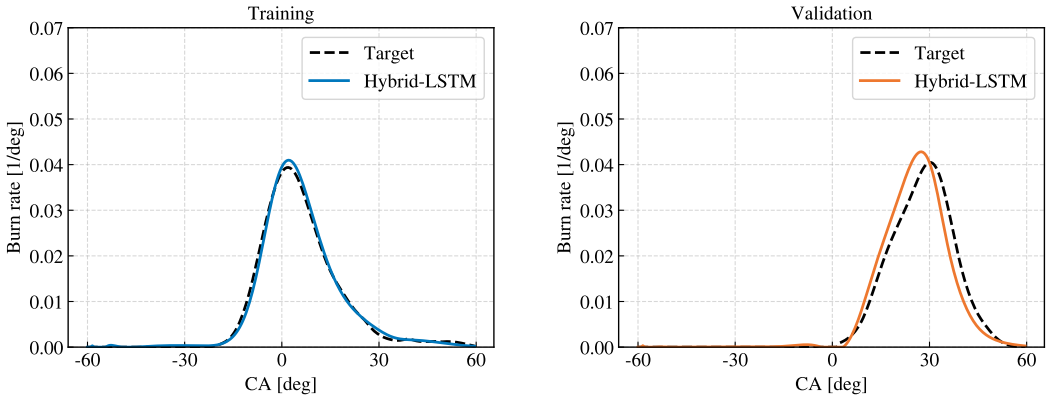


Fig. 4.12 Training and validation loss versus epoch. Both curves decrease smoothly and converge to closely matched stationary values, indicating stable learning and a negligible generalization gap.



(a) (b)

Fig. 4.13 Representative burn rate profiles reconstructed by the Hybrid-LSTM. (a) Training cycle with high-fidelity fit. (b) Validation cycle with strongly delayed combustion (rare in the dataset) accurately captured.

is stable and generalizes well across operating conditions, including challenging delayed-combustion cases.

4.4.2 MLP for Wiebe Parameters Prediction

This section evaluates a feed-forward NN trained to predict single-Wiebe parameters whose targets are obtained from GA fits as explained in Section 4.3.3. The model thus learns a mapping from operating conditions to GA-optimized coefficients and reconstructs the burn rate profile from the inferred parameters. The following results illustrate its behavior across representative sweeps.

Across the speed load sweep under stoichiometric and no-EGR conditions, and λ–EGR sweeps at 1500rpm $\times$ 5.5bar BMEP. The network reproduces the overall burn rate shape with good fidelity: the central portion of the burn is well captured, but two systematic effects emerge, (i) a tendency to overshoot the peak burn rate and (ii) reduced accuracy at the very beginning and end of combustion. In particular, the onset is often delayed, with predicted MFB10 occurring a few crank-angle degrees later than measured, while the decay tail can be truncated. These behaviors are visible both in the stoichiometric, no-EGR speed–load sweep (Figure 4.14) and in the λ/EGR sweep at 1500rpm (Figure 4.15), where the model tracks the main evolution of the MFB but exhibits localized deviations at the start and tail of combustion.

Consistent patterns arise in the combustion metrics. The center of combustion (MFB50) is generally well matched, whereas the combustion duration (MFB10–75) is systematically underestimated. Despite these biases, most points remain within a $\pm$5 CA error band relative to the experimental references, indicating that the model captures the principal trends across operating points; this is summarized by the global regressions in Figure 4.16. Overall, the network produces physically plausible MFB profiles over the explored λ/EGR conditions, with remaining discrepancies concentrated at the onset and tail of combustion. Further insights regarding the model performance can be found in the Appendix A.1.

4.4.3 Hybrid Recurrent Model for Burn Rate Prediction

To overcome the limitations imposed by the S-shaped structure of the Wiebe function for the MFB evolution, an hybrid recurrent NN–based model is adopted, as described in Section 4.3.4. After assessing training performance in Section 4.4.1, it is essential to verify whether the hybrid–recurrent model behaves robustly across operating

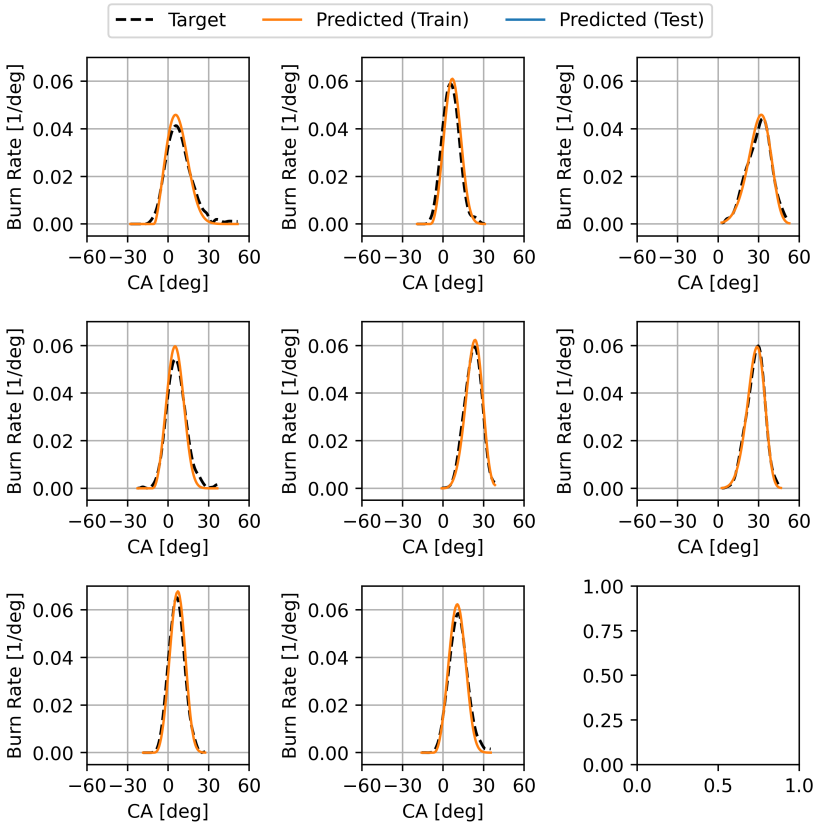


Fig. 4.14 Wiebe-parameter MLP across the stoichiometric, no-EGR speed–load sweep. The model captures the central burn region but shows peak overshoot and localized discrepancies at onset and tail.

conditions and captures the changes in combustion evolution induced by speed, load, and dilution, in line to the results presented for the Wiebe-NN model. Accordingly, results are presented through three operating condition sweeps:dilution (EGR–λ), speed, and load, reflecting the data-collection plan in Section 4.2. Finally, the model's generalization over the entire dataset is evaluated using the combustion metrics employed throughout this chapter (MFB10, MFB50, and MFB10–75).

Performance Across Operating Conditions

This subsection examines the model's physical consistency and generalization beyond the training set by probing operating regimes not contiguous in feature space. For each sweep, panels are arranged to highlight changes in the independent vari-

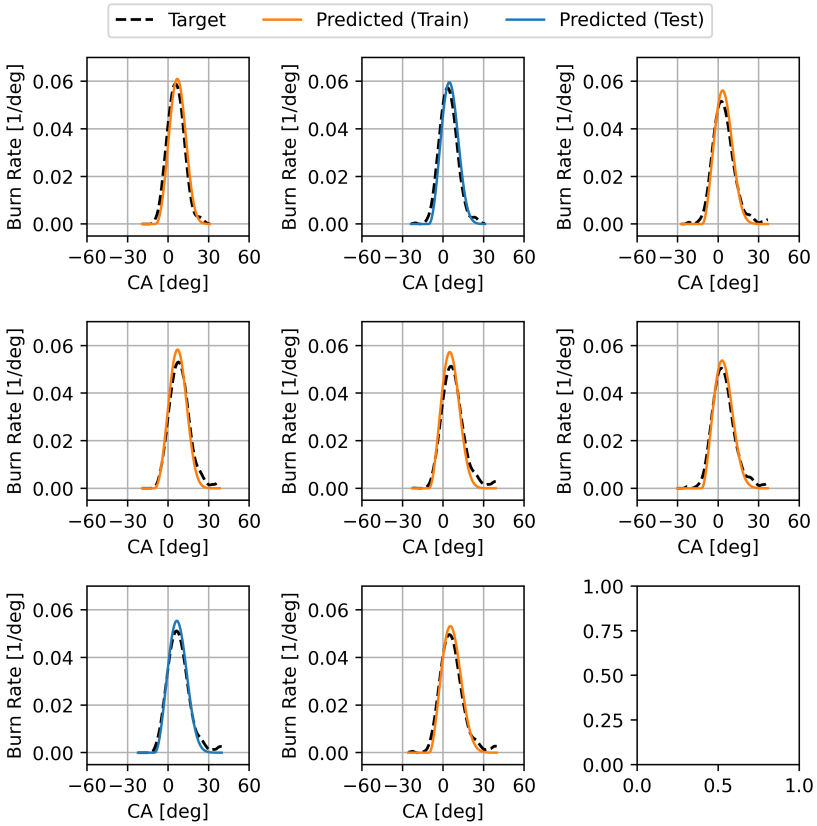


Fig. 4.15 Wiebe-parameter MLP for the λ–EGR sweep at 1500rpm $\times$ 5.5bar BMEP. The overall shape is reproduced, with typical onset delay (MFB10) and occasional truncation of the decay tail.

ables, allowing a visual check that predicted burn rate shapes evolve plausibly with the underlying physics.

Speed and Load Effect Figure 4.17 reports the speed–load sweep (speed increasing from top to bottom, load from left to right) under stoichiometric, no-EGR conditions. The hybrid–recurrent model reproduces the burn rate evolution with high fidelity: combustion duration, peak intensity, and peak position closely match the references, with no evident differences among training, validation, and test panels, an indication of good generalization. A mild loss of accuracy appears only in cases with delayed combustion phasing (slightly underestimated peak intensity), which are under-represented in the dataset. Compared with the MLP–Wiebe baseline in Figure 4.14, the hybrid–recurrent model mitigates the typical peak overshoot and the

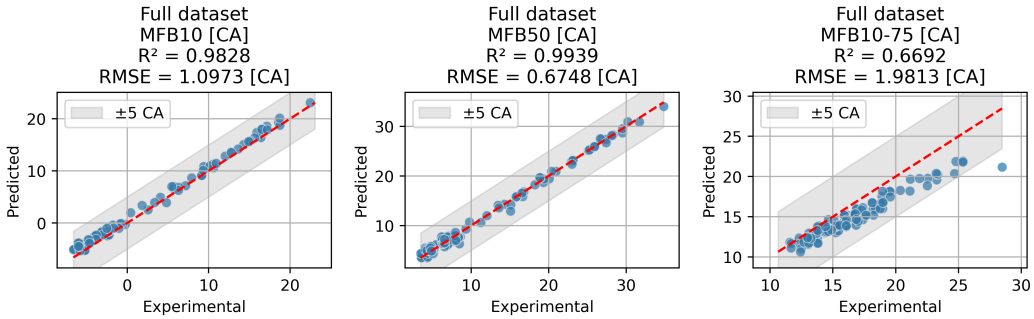


Fig. 4.16 Regression of predicted versus experimental combustion metrics for the Wiebe-parameter MLP across the full dataset. Most points lie within ±5 CA, with a tendency to underestimate MFB10–75.

onset/tail biases of the Wiebe reconstruction (delayed MFB10 and truncated decay), while preserving the correct peak location across the map.

Improving performance in the delayed-combustion regime is expected by enriching the operating-point distribution with additional late-phasing cases.

Dilution Effect Figure 4.18 summarizes the dilution sweep at 1500 rpm and 5.5 bar IMEP, with EGR increasing from top to bottom and λ increasing from left to right. The hybrid–recurrent model tracks the systematic effect of dilution on the burn rate shape, reproducing the broadening of the curve and the slight delay of the peak as EGR rises, as well as the reduction in peak intensity and modest phasing delay as the mixture becomes leaner. Combustion duration, peak magnitude, and peak location remain in close agreement with the targets across all data splits.

Very similar behavior is observed in Figure A.4a, which shows the same EGR–λ sweep at a higher speed–load point (3000 rpm, 7 bar IMEP), again ordered with EGR from top to bottom and λ from left to right. The model accurately follows the expected dilution trends—widening burns, attenuated peaks, and a shift of the peak to later crank angles as EGR or λ increase, with uniformly strong agreement across training, validation, and test subsets.

Taken together across the two dilution sweeps (1500 rpm–5.5 bar and 3000 rpm–7 bar), the hybrid–recurrent model consistently reproduces the expected effects of increasing EGR and λ—broader burns, attenuated peaks, and slight phasing delays, while maintaining close agreement with targets for all data splits; relative

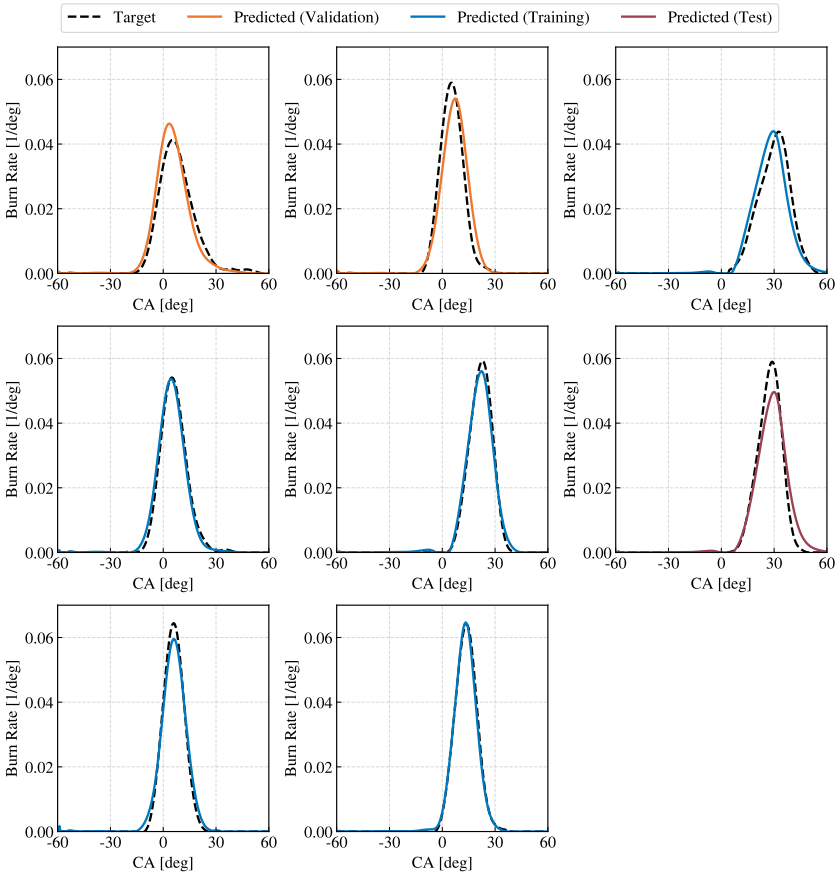


Fig. 4.17 Comparison between target and Hybrid-LSTM predicted burn rate traces over a speed–load sweep under stoichiometric, no-EGR conditions ($\lambda = 1$, $EGR = 0$). Engine speed increases from top to bottom, while load increases from left to right. The black dashed lines denote the target; orange, blue, and red lines indicate predictions on validation, training, and test samples, respectively.

to the MLP–Wiebe baseline, it reduces the onset delay, peak overshoot, and end-of-combustion truncation typical of the Wiebe reconstruction. For a condensed view of metric evolution across operating conditions, an aggregated summary is provided in Appendix A.1.

Combustion Metrics

Figure 4.20 summarizes the generalization performance over the full dataset for the combustion metrics considered in this chapter (MFB10, MFB50, and MFB10–75). Data from the training, validation, and test splits cluster closely around the 1:1 line

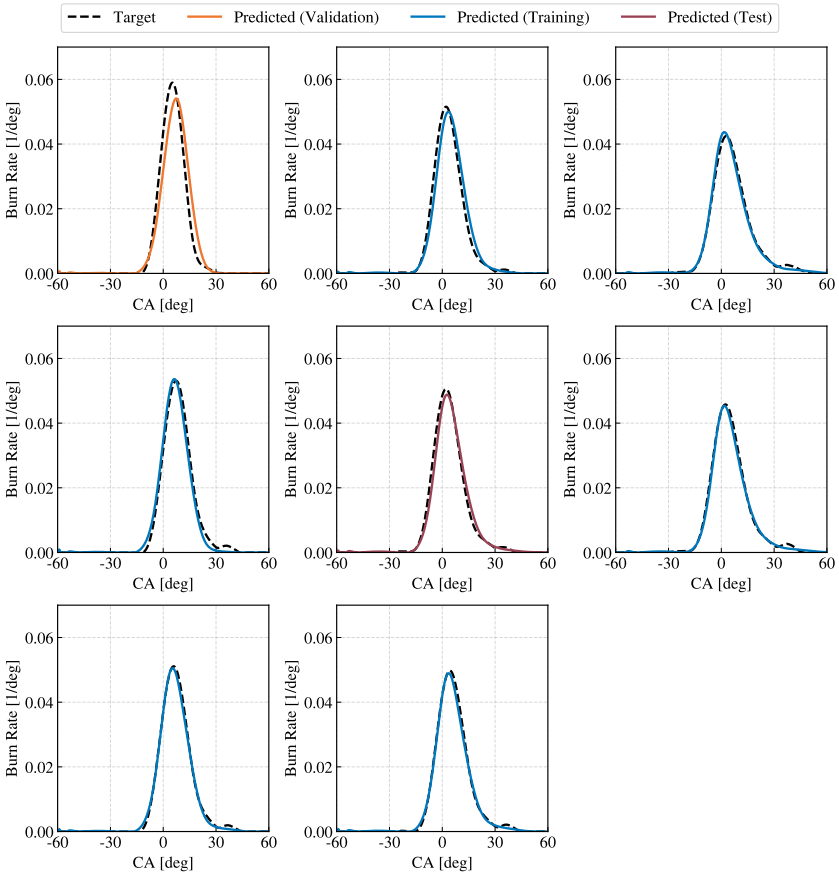


Fig. 4.18 Hybrid-LSTM burn rate predictions for an EGR–λ sweep at fixed operating point: $n = 1500$rpm, $IMEP = 5.5$bar. Panels are ordered with EGR increasing from top to bottom and λ increasing from left to right. Dashed black lines are targets; solid blue and maroon lines are predictions on training and test samples, respectively, and orange lines (when present) denote validation.

and mostly fall within the shaded ± 5 CA band, yielding low RMSE values and high coefficients of determination (R^2).

When comparing the LSTM-based reconstruction with the Wiebe-parameter approach, different trends emerge across metrics. In the Wiebe-based model, combustion phasing (MFB50) is directly embedded in the parametric formulation, and the calibration procedure, based on a genetic algorithm, was biased toward minimizing phasing errors, even at the expense of less accurate predictions of combustion start and tail. This makes MFB50 inherently easier to predict within the Wiebe framework. Conversely, the LSTM model is required to reconstruct the entire combustion evolution without explicitly enforcing phasing as a parameter, making the

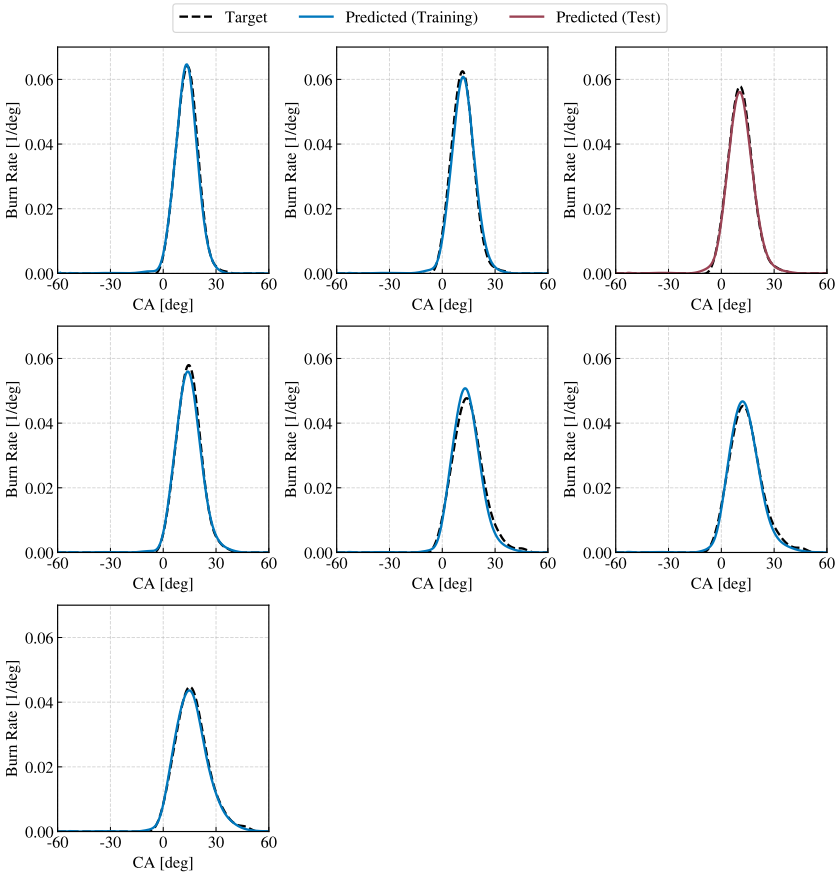


Fig. 4.19 Hybrid-LSTM burn rate predictions for an EGR–λ sweep at fixed operating point: $n = 3000$rpm, $IMEP = 13$bar. Panels are ordered with EGR increasing from top to bottom and λ increasing from left to right. Dashed black lines are targets; solid blue and maroon lines are predictions on training and test samples, respectively, and orange lines (when present) denote validation.

accurate prediction of MFB50 intrinsically more challenging. Despite this, the LSTM approach achieves excellent accuracy in terms of combustion phasing, with a small deviation from the value tracked by the Wiebe-based model (see Figure 4.16), while providing a markedly improved reconstruction of combustion start and duration (MFB10 and MFB10–75). This highlights a fundamental trade-off between parametric simplicity and global combustion fidelity: the LSTM-based approach entails increased modeling complexity but yields a more balanced and physically consistent description of the combustion process.

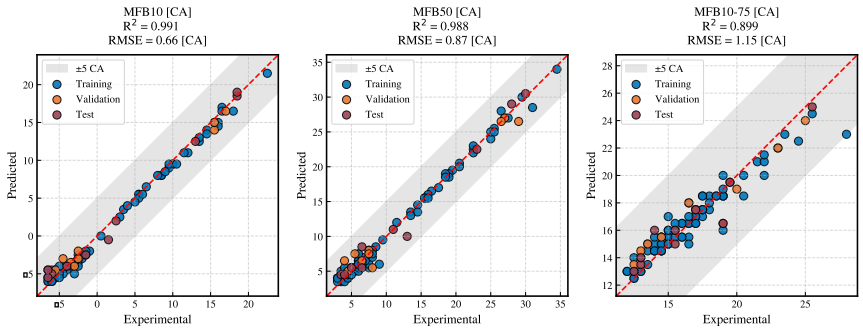


Fig. 4.20 Regression of predicted versus experimental combustion metrics over the full dataset. Shaded band indicates ±5 CA around the 1:1 line; colors denote train/validation/test.

4.5 Conclusion and Future Steps

The Hybrid-LSTM reconstructs burn rate profiles with high fidelity across speed, load, and dilution; training and validation losses converge to similar stationary values, and combustion metrics cluster around the 1:1 line (RMSEs of 0.81 CA for MFB10, 0.93 CA for MFB50, and 1.07 CA for MFB10–75). Compared with the MLP–Wiebe baseline, the model mitigates onset delay, peak overshoot, and tail truncation while preserving peak phasing; the residual bias is concentrated in rare late-phasing, long-duration cases, indicating that data coverage is the main limiter. Next steps prioritize expanding and rebalancing the dataset by adding operating points and systematically sweeping key actuations (e.g., spark advance, injection timing/strategy, boost/throttle), as well as incorporating transient maneuvers to capture dynamic effects. In parallel, integrating the data-driven combustion model into a GT-POWER engine test bench will enable benchmarking in a comprehensive co-simulation environment and quantification of computational speed-ups relative to phenomenological combustion models, establishing a practical speed–accuracy trade-off for calibration, virtual testing workflows, and real-time engine-model operation.

Chapter 5

Conclusion

In an era where climate change is the defining engineering challenge, every incremental cut in emissions counts. In parallel, the digital and AI revolution provides practical tools, such as high-fidelity data, scalable computation, and connected infrastructures, that make it possible to sense, predict, and coordinate complex mobility and powertrain systems across scales. This dissertation leverages these tools through domain knowledge-aware, learning-based controllers and data-driven models that embed physics and optimization references to preserve feasibility and interpretability, while meeting real-time inference constraints for real-world deployment.

From a cross-scale perspective, several principles emerge. Combining prior knowledge, whether physical laws or control structure, with learning converts raw computation into reliable action: convex dispatch layers, costed flows, and physics-informed losses stabilize training, while reinforcement learning and sequence models capture variability that short-horizon optimizers or rigid parametric laws miss. Representation matters: graphs for spatial coupling and sequences for temporal dependencies act as inductive biases that steer learning toward physically plausible behavior. Evaluation must reflect operations: KPIs that combine service quality, utilization, emissions, energy/fuel use, and combustion metrics expose real trade-offs rather than only average errors. Finally, broad, scenario-relevant coverage is essential so models interpolate rather than extrapolate when stressed.

Building on these principles, the city-scale AMoD layer formulates fleet control as a network-flow problem. A graph-encoded SAC policy proposes region-level allocations, and a minimum cost flow layer enforces feasibility. In a calibrated

mesoscopic SUMO model of Luxembourg, this hierarchy achieved near-MPC profit, reduced passenger waiting times, and contained empty kilometers, lowering pollutant and CO_2 emissions. Gains arise from when and where relocations occur, underscoring the value of predictive rather than reactive rebalancing.

At the powertrain scale, learning-based energy management on a plug-in hybrid reaches close to Dynamic Programming trends for CO_2 and terminal energy constraints and outperforms ECMS baselines while being real-time implementable. Policies met state of charge targets in both charge sustaining and charge depleting operation. Closed-loop validation on a detailed vehicle virtual test rig across heterogeneous cycles and initial SoC levels, together with interpretability maps over states, clarified how torque split and battery usage are traded.

At the combustion scale, two models were developed: an MLP that outputs single-Wiebe parameters and a hybrid LSTM that predicts the crank angle–resolved burn rate profile. The Wiebe-MLP captured mid-burn combustion phasing but tended to delay start and truncate the tail, whereas the Hybrid-LSTM mitigated these issues, preserved peak phasing, and remained stable across splits and sweeps in speed, load, and dilution.

Looking forward, broader applicability and robustness call for tighter cross-scale coupling and richer learning signals. A city–vehicle closed loop that exposes trip progress and congestion forecasts to the EMS would allow torque split and battery usage to respond to network-level guidance, enabling a more accurate quantification of fleet-level energy and emissions gains; adding speed-forecasting layers inside the EMS can reduce partial observability and further narrow the gap to DP-like performance. Integrating the predictive combustion surrogate into the hybrid vehicle's virtual test rig can raise ICE fidelity without extra computational cost, an essential constraint for learning-based controllers that phenomenological turbulence/combustion models struggle to meet. Extending interpretability from EMS to AMoD and combustion, via statistical attribution, targeted visualization, and explainable machine learning, can diagnose policy decisions and surrogate predictions. Dataset enrichment is equally important: wider city scenarios for AMoD and EMS, and expanded engine operating envelopes for combustion, will stress-test edge regimes and promote interpolation instead of extrapolation, increasing applicability and generalization of data-driven models. Finally, exploring meta-RL and offline RL

schemes, together with more explicitly physics-aware losses and architectures, can improve data efficiency, stability, and deployability.

Taken together, the results show that learning controllers, when grounded in domain structure and benchmarked against strong optimization baselines, can deliver real-time decisions that scale from city networks to embedded control units and into combustion model surrogates, advancing the practical integration of AI with energy-efficient mobility. By aligning representations with physics and evaluation with operations, the methodology narrows the gap between algorithmic promise and deployable practice. The remaining challenges, like broader data coverage in edge regimes, tighter city–vehicle coupling, and richer physics-aware learning signals, are tractable and outline a clear roadmap for scale-up.

References

[1] Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco C. Pereira, and Marco Pavone. Graph neural network reinforcement learning for autonomous mobility-on-demand systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 2996–3003, 2021.

[2] Luigi Tresca, Luca Pulvirenti, and Luciano Rolando. A Cutting-Edge Energy Management System for a Hybrid Electric Vehicle relying on Soft Actor–Critic Deep Reinforcement Learning. *Transportation Engineering*, 19:100308, 3 2025.

[3] A. Onorati and G. Montenegro. *1D and Multi-D Modeling Techniques for IC Engine Simulation*. SAE International, 2020.

[4] J. Heywood. *Internal Combustion Engine Fundamentals*. McGraw-Hill Education, 1988.

[5] T. Poinsot and D. Veynante. *Theoretical and Numerical Combustion*. Edwards, 2005.

[6] UNECE. European green deal: commission proposes transformation of eu economy and society to meet climate ambitions," accessed october 2021. https://ec.europa.eu/commission/presscorner/detail/en/IP_21_3541., 2021, online documentation.

[7] IEA. Largest end uses of energy by sector in selected iea countries, 2019. https://www.iea.org/data-and-statistics/charts/largest-end-uses-of-energy-by-sector-in-selected-iea-countries-2019., 2022, online documentation.

[8] UN Dep. Econ. Soc. Aff. 68% of the world population projected to live in urban areas by 2050, 2021. Available at un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.

[9] OECD. The cost of air pollution - health impacts of road transport. Technical report, 2014.

[10] C. Kuhnimhof, T.and Eisenmann. Mobility-on-demand pricing versus private vehicle tco: how cost structures hinder the dethroning of the car. *Transportation*, 50, 20123.

[11] G. Zardini, N. Lanzetti, M. Pavone, and E. Frazzoli. Analysis and control of autonomous mobility-on-demand systems: A review. 2022.

[12] S. Oh, A. F. Lentzakis, R. Seshadri, and M. Ben-Akiva. Impacts of automated mobility-on-demand on traffic dynamics, energy and emissions: A case study of singapore. *Simulation Modelling Practice and Theory*, 2021.

[13] ACEA. Electric vehicles: tax benefits and purchase incentives. https://www.acea.auto/files/Electric_vehicles-Tax_benefits_purchase_incentives_European_Union_2020.pdf., 2021, online documentation.

[14] EEA. New registrations of electric vehicles in europe. hhttps://www.eea.europa.eu/ims/new-registrations-of-electric-vehicles., 2022, online documentation.

[15] IEA. Global electric car sales have continued their strong growth in 2022 after breaking records last year. https://rb.gy/3wf5c., 2022, online documentation.

[16] Lars-Henrik Björnsson and Sten Karlsson. Electrification of the two-car household: Phev or bev? *Transportation Research Part C: Emerging Technologies*, 85:363–376, 2017.

[17] Antonio Sciarretta and Lino Guzzella. Control of hybrid electric vehicles. *IEEE Control Systems Magazine*, pages 60–70, April 2007.

[18] Dai-Duong Tran, Majid Vafaeipour, Mohamed El Baghdadi, Ricardo Barrero, Joeri Van Mierlo, and Omar Hegazy. Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: Topologies and integrated energy management strategies. *Renewable and Sustainable Energy Reviews*, 119:109596, 2020.

[19] Atriya Biswas and Ali Emadi. Energy management systems for electrified powertrains: State-of-the-art review and future trends. *IEEE Transactions on Vehicular Technology*, 68(7):6453–6467, 2019.

[20] Antonio Riccio, Filippo Monzani, and Maurizio Landi. Towards a powerful hardware-in-the-loop system for virtual calibration of an off-road diesel engine. *Energies*, 15(2), 2022.

[21] Sebastian Grasreiner, Jens Neumann, Michael Wensing, and Christian Hasse. Model-based virtual engine calibration with the help of phenomenological methods for spark-ignited engines. *Applied Thermal Engineering*, 121:190–199, 2017.

[22] Yingcong Zhou, Brian Gainey, and Benjamin Lawler. An ultrafast multizone hcci model with autoignition, global reaction and interpolation (agi) for achieving comparable accuracy to detailed chemical kinetics models. *Combustion and Flame*, 221:487–501, 2020.

[23] S. Posch, C. Gößnitzer, M. Lang, R. Novella, H. Steiner, and A. Wimmer. Turbulent combustion modeling for internal combustion engine cfd: A review. *Progress in Energy and Combustion Science*, 106:101200, 2025.

[24] Shihong Zhang, Chi Zhang, and Bosen Wang. Crk-pinn: A physics-informed neural network for solving combustion reaction kinetics ordinary differential equations. *Combustion and Flame*, 269:113647, 2024.

[25] Daniele Gammelli, Inon Peled, Filipe Rodrigues, Dario Pacino, Haci A. Kurtaran, and Francisco C. Pereira. Estimating latent demand of shared mobility through censored gaussian processes. *Transportation Research Part C: Emerging Technologies*, 120:102775, 2020.

[26] Carolin Schmidt, Daniele Gammelli, Francisco Camara Pereira, and Filipe Rodrigues. Learning to control autonomous fleets from observation via offline reinforcement learning. In *2024 European Control Conference (ECC)*, pages 1399–1406, 2024.

[27] Markus Frey, Dirk Itzen, Johannes Sautter, Louis Weller, Timo Hagenbucher, Qirui Yang, Michael Grill, and Andre Casal Kulzer. Development of ultra-fast ai-driven diesel engine model for real-time optimization. *SAE Technical Papers*, 4 2025.

[28] Hao Yuan, Harsh Goyal, Reza Islam, Karl Giles, Simeon Howson, Andrew Lewis, Dom Parsons, Stefania Esposito, Sam Akehurst, Peter Jones, Matthew McAllister, Bryn Littlefair, Zhewen Lu, and Sipeng Zhu. Thermodynamics-based data-driven combustion modelling for modern spark-ignition engines. *Energy*, 313, 12 2024.

[29] Nils J. Nilsson. *The Quest for Artificial Intelligence*. Cambridge University Press, 2009.

[30] Susmita Ray. A quick review of machine learning algorithms. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 35–39, 2019.

[31] Diksha Sharma and Neeraj Kumar. A review on machine learning algorithms, tasks and applications. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 6(10):2278–1323, 2017.

[32] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, PP:1–1, 04 2019.

[33] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[34] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[35] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[36] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[37] Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. Deep learning on traffic prediction: Methods, analysis, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4927–4943, 2022.

[38] Federico Millo, Luciano Rolando, Luigi Tresca, and Luca Pulvirenti. Development of a neural network-based energy management system for a plug-in hybrid electric vehicle. *Transportation Engineering*, 11, 3 2023.

[39] Huifang Kong, Yao Fang, Lei Fan, Hai Wang, Xiaoxue Zhang, and Jie Hu. A novel torque distribution strategy based on deep recurrent neural network for parallel hybrid electric vehicle. *IEEE Access*, 7:65174–65185, 2019.

[40] Bin Xu, Dhruvang Rathod, Darui Zhang, Adamu Yebi, Xueyu Zhang, Xiaoya Li, and Zoran Filipi. Parametric study on reinforcement learning optimized energy management strategy for a hybrid electric vehicle. *Applied Energy*, 259, 2 2020.

[41] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[42] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[43] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[45] Paul Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science*. PhD thesis, Appl. Math. Harvard University, 01 1974.

[46] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2013.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[48] Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. pages 324–328, 11 2016.

[49] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[50] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[51] Fan Zhou, Qing Yang, Ting Zhong, Dajiang Chen, and Ning Zhang. Variational graph neural networks for road traffic prediction in intelligent transportation systems. *IEEE Transactions on Industrial Informatics*, 17:2802–2812, 4 2021.

[52] Saeed Rahmani, Asiye Baghbani, Nizar Bouguila, and Zachary Patterson. Graph neural networks for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 24:8846–8885, 8 2023.

[53] D. Gammelli, J. Harrison, K. Yang, M. Pavone, F. Rodrigues, and Pereira C. Francisco. Graph reinforcement learning for network control via bi-level optimization. In *Int. Conf. on Machine Learning*, 2023.

[54] Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco Pereira, and Marco Pavone. Graph meta-reinforcement learning for transferable autonomous mobility-on-demand. pages 2913–2923. Association for Computing Machinery, 8 2022.

[55] Microscopic traffic simulation using sumo. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-November:2575–2582, 12 2018.

[56] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018.

[57] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2:160–163, 7 1991.

[58] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 6 2019.

[59] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *8th International Conference on Learning Representations, ICLR 2020*, 12 2019.

[60] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning 1992 8:3*, 8:279–292, 5 1992.

[61] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[62] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 9 2015.

[63] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov Openai. Proximal policy optimization algorithms. 7 2017.

[64] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. *32nd International Conference on Machine Learning, ICML 2015*, 3:1889–1897, 2 2015.

[65] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[66] Hado Van Hasselt. Double q-learning. *Advances in Neural Information Processing Systems*, 23, 2010.

[67] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *CoRR*, cs.AI/9605103, 1996.

[68] N. Metropolis and S. Ulam. The monte carlo method. *J. Am. Stat. Assoc.*, 44:335, 1949.

[69] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

[70] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1 – 26, 1979.

[71] M. Huneault and F.D. Galiana. A survey of the optimal power flow literature. *IEEE Transactions on Power Systems*, 1991.

[72] V. Popovskij, A. Barkalov, and L. Titarenko. *Control and Adaptation in Telecommunication Systems - Mathematical Foundations*, volume 94. 01 2011.

[73] M. A. Bellamy and R. C. Basole. Network analysis of supply chain systems: A systematic review and future research. *Systems Engineering*, 2013.

[74] Yi Wang, W.Y. Szeto, Ke Han, and Terry L. Friesz. Dynamic traffic assignment: A review of the methodological advances for environmentally sustainable road transportation applications. *Transportation Research Part B: Methodological*, 2018.

[75] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 1958.

[76] G. B. Dantzig. Reminiscences about the origins of linear programming. *Operations Research Letters*, 1982.

[77] M. Hyland and H.-S. Mahmassani. Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests. 2018.

[78] Z. Liu, T. Miwa, W. Zeng, M. G.H. Bell, and T. Morikawa. Dynamic shared autonomous taxi system considering on-time arrival reliability. *Transportation Research Part C: Emerging Technologies*, 2019.

[79] R. Zhang and M. Pavone. Control of robotic Mobility-on-Demand systems: A queueing-theoretical perspective. *Int. Journal of Robotics Research*, 2016.

[80] R. Zhang, F. Rossi, and M. Pavone. Model predictive control of Autonomous Mobility-on-Demand systems. In *Proc. IEEE Conf. on Robotics and Automation*, Stockholm, Sweden, May 2016.

[81] C. Mao, Y. Liu, and Z. Shen. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 2020.

[82] J. Feng, M. Gluzman, and J. G. Dai. Scalable deep reinforcement learning for ride-hailing*. *Proc. of the American Control Conf.*, 2021.

[83] S. He, Y. Wang, S. Han, S. Zou, and F. Miao. A robust and constrained multi-agent reinforcement learning electric vehicle rebalancing method in amod systems. *IEEE Int. Conf. on Intelligent Robots and Systems*, 2023.

[84] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[85] Theo Hofman, Maarten Steinbuch, Roell Van Druten, and Alex Serrarens. Rule-based energy management strategies for hybrid vehicles. *International Journal of Electric and Hybrid Vehicles*, 1(1):71–94, 2007.

[86] H. Kaufman. The mathematical theory of optimal processes, by l. s. pontryagin, v. g. boltyanskii, r. v. gamkrelidze, and e. f. mishchenko. authorized translation from the russian. translator: K. n. trirogoff, editor: L.. w. neustadt. interscience publishers (division of john wiley and sons, inc., new york) 1962. viii 360 pages. *Canadian Mathematical Bulletin*, 7(3):500–500, 1964.

[87] Gino Paganelli, Thierry-Marie Guerra, Sebastien Delprat, Jean-Jacques Santin, M. Delhom, and E. Combes. Simulation and assessment of power control strategies for a parallel hybrid car. *Int. J. of Automobile Engineering*, 214:705–717, 07 2000.

[88] Basil Kouvaritakis and Mark Cannon. *Model Predictive Control: Classical, Robust and Stochastic*. Springer, 01 2016.

[89] Luca Pulvirenti, Luciano Rolando, and Federico Millo. Energy management system optimization based on an lstm deep learning model using vehicle speed prediction. *Transportation Engineering*, 11:100160, 2023.

[90] Alessia Musa, Pier Giuseppe Anselma, Giovanni Belingardi, and Daniela Anna Misul. Energy management in hybrid electric vehicles: A q-learning solution for enhanced drivability and energy efficiency. *Energies*, 17(1):62, 2023.

[91] Saeid Ahmadian, Mohammad Tahmasbi, and Reza Abedi. Q-learning based control for energy management of series-parallel hybrid vehicles with balanced fuel consumption and battery life. *Energy and AI*, 11:100217, 2023.

[92] Bin Shuai, Quan Zhou, Ji Li, Yinglong He, Ziyang Li, Huw Williams, Hongming Xu, and Shijin Shuai. Heuristic action execution for energy efficient charge-sustaining control of connected hybrid vehicles with model-free double q-learning. *Applied energy*, 267:114900, 2020.

[93] Jingda Wu, Hongwen He, Jiankun Peng, Yuecheng Li, and Zhanjiang Li. Continuous reinforcement learning of energy management with deep q network for a power split hybrid electric bus. *Applied energy*, 222:799–811, 2018.

[94] Xuefeng Han, Hongwen He, Jingda Wu, Jiankun Peng, and Yuecheng Li. Energy management based on reinforcement learning with double deep q-learning for a hybrid electric tracked vehicle. *Applied Energy*, 254:113708, 2019.

[95] Luigi Tresca, Luca Pulvirenti, Luciano Rolando, and Federico Millo. Development of a deep q-learning energy management system for a hybrid electric vehicle. *Transportation Engineering*, 16:100241, 2024.

[96] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *35th International Conference on Machine Learning, ICML 2018*, 4:2587–2601, 2 2018.

[97] Renzong Lian, Jiankun Peng, Yuankai Wu, Huachun Tan, and Hailong Zhang. Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle. *Energy*, 197:117297, 2020.

[98] Xuanang Zhang, Xuan Wang, Ping Yuan, Hua Tian, and Gequn Shu. Optimizing hybrid electric vehicle coupling organic rankine cycle energy management strategy via deep reinforcement learning. *Energy and AI*, 17:100392, 2024.

[99] Ruchen Huang, Hongwen He, Xuyang Zhao, Yunlong Wang, and Menglin Li. Battery health-aware and naturalistic data-driven energy management for hybrid electric bus based on td3 deep reinforcement learning algorithm. *Applied Energy*, 321:119353, 2022.

[100] Jianhao Zhou, Siwu Xue, Yuan Xue, Yuhui Liao, Jun Liu, and Wanzhong Zhao. A novel energy management strategy of hybrid electric vehicle via an improved td3 deep reinforcement learning. *Energy*, 224:120118, 2021.

[101] Yang Lin, Liang Chu, Jincheng Hu, Zhuoran Hou, Jihao Li, Jingjing Jiang, and Yuanjian Zhang. Progress and summary of reinforcement learning on energy management of mps-ev. *Heliyon*, 2023.

[102] Weiwei Huo, Tianyu Zhao, Fan Yang, and Yong Chen. An improved soft actor-critic based energy management strategy of fuel cell hybrid electric vehicle. *Journal of Energy Storage*, 72:108243, 2023.

[103] Weiqi Chen, Jiankun Peng, Jun Chen, Jiaxuan Zhou, Zhongbao Wei, and Chunye Ma. Health-considered energy management strategy for fuel cell hybrid electric vehicle based on improved soft actor critic algorithm adopted with beta policy. *Energy Conversion and Management*, 292:117362, 2023.

[104] Luciano Rolando, Nicola Campanelli, Luigi Tresca, Luca Pulvirenti, and Federico Millo. Development of a soft-actor critic reinforcement learning algorithm for the energy management of a hybrid electric vehicle. Technical report, SAE Technical Paper, 2024.

[105] Luigi Tresca, Carolin Schmidt, James Harrison, Filipe Rodrigues, Gioele Zardini, Daniele Gammelli, and Marco Pavone. Robo-taxi fleet coordination at scale via reinforcement learning, 2025.

[106] Luigi Tresca, Luca Pulvirenti, and Luciano Rolando. The effect of the initial battery state of charge on the performance of a soft actor-critic energy management system. 9 2025.

[107] G.F. Newell. A simplified theory of kinematic waves in highway traffic, part i: General theory. *Transportation Research Part B: Methodological*, 27(4):281–287, 1993.

[108] J. Lighthill and G.B. Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229:317–345, 5 1955.

[109] Paul I. Richards. Shock waves on the highway. *https://doi.org/10.1287/opre.4.1.42*, 4:42–51, 2 1956.

[110] Carlos F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, August 1994.

[111] V.. Sundarapandian. Probability, statistics and queuing theory. page 809, 2009.

[112] S. Krauss, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55:5597, 5 1997.

[113] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221–2229, 12 1992.

[114] Arne Kesting, Martin Treiber, and Dirk Helbing. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605, October 2010.

[115] General lane-changing model mobil for car-following models. *Transportation Research Record*, pages 86–94, 2007.

[116] Federico Millo, Luciano Rolando, and Maurizio Andreata. Numerical simulation for vehicle powertrain development. In Jan Awrejcewicz, editor, *Numerical Analysis*, chapter 24. IntechOpen, Rijeka, 2011.

[117] Dimitri Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 1995.

[118] Johannes Van Der Wal. *Stochastic dynamic programming*. PhD thesis, Methematisch Centrum Amsterdam, The Netherlands, 1980.

[119] Philipp Valentin Elbert, Soeren Ebbesen, and Lino Guzzella. Implementation of dynamic programming for n-dimensional optimal control problems with final state constraints. *IEEE Transactions on Control Systems Technology*, 21, 2013-05.

[120] Hans Geering. *Optima Control with Engineering Applications*. Springer, Berlin, 01 2007.

[121] Gino Paganelli. *Conception et commande d'une chaîne de traction pour véhicule hybride parallèle thermique et électrique*. Université de Valenciennes et du Hainaut Cambrésis, 1999.

[122] Lorenzo Serrao, Simona Onori, and Giorgio Rizzoni. Ecms as a realization of pontryagin's minimum principle for hev control. In *2009 American Control Conference*, pages 3964–3969, 2009.

[123] Pierluigi Pisu and Giorgio Rizzoni. A comparative study of supervisory control strategies for hybrid electric vehicles. *Control Systems Technology, IEEE Transactions on*, 15(3):506–518, 2007.

[124] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25:335–348, 5 1989.

[125] Hoseinali Borhan, Ardalan Vahidi, Anthony M. Phillips, Ming L. Kuang, Ilya V. Kolmanovsky, and Stefano Di Cairano. Mpc-based energy management of a power-split hybrid electric vehicle. *IEEE Transactions on Control Systems Technology*, 20:593–603, 5 2012.

[126] Hongwen He, Yunlong Wang, Ruoyan Han, Mo Han, Yunfei Bai, and Qingwu Liu. An improved mpc-based energy management strategy for hybrid vehicles using v2v and v2i communications. *Energy*, 225:120273, 6 2021.

[127] Rick Zhang, Federico Rossi, and Marco Pavone. Model predictive control of autonomous mobility-on-demand systems. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1382–1389, 9 2015.

[128] Z. Lei, X. Qian, and S.-V. Ukkusuri. Efficient proactive vehicle relocation for on-demand mobility service with recurrent neural networks. 117, 2020.

[129] C. Ruch, J. Gächter, J. Hakenberg, and E. Frazzoli. The+ 1 method: model-free adaptive repositioning policies for robotic multi-agent systems. *IEEE Transactions on Network Science and Engineering*, 2020.

[130] Chan-Chiao Lin, Huei Peng, JW Grizzle, and Jun-Mo Kang. Power management strategy for a parallel hybrid electric truck. *IEEE Transactions on Control Systems Technology*, 11(6):839–849, 2003.

[131] Domenico Bianchi, Luciano Rolando, Lorenzo Serrao, Simona Onori, Giorgio Rizzoni, Nazhar Al-Khayat, Tung-Ming Hsieh, and Pengju Kang. A rule-based strategies for a series-parallel hybrid electric vehicle: an approach based on dynamic programming. *Proceedings of The ASME Dynamic Systems and Control Conference, Cambridge, Massachusetts*, September 2010.

[132] F. Rossi, R. Zhang, Y. Hindy, and M. Pavone. Routing autonomous vehicles in congested transportation networks: Structural properties and coordination algorithms. 42(7):1427–1442, 2018.

[133] Lara Codeca, Raphael Frank, Sebastien Faye, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 6 2017.

[134] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2:1–21, 5 2021.

[135] Johannes Claßen, Stefan Sterlepper, Frank Dorscheidt, Michael Görgen, Johannes Scharf, Martin Nijs, Norbert Alt, Andreas Balazs, Marius Böhmer, Matthieu Doucet, Sascha Krysmon, and Christian Boßer. Rde cycle generation – a statistical approach to cut down testing effort and provide a secure base to approve rde legislation compliance. pages 37–56, 2019.

[136] Jelica Pavlovic, Alessandro Tansini, Jaime Suarez, and Georgios Fontaras. Influence of vehicle and battery ageing and driving modes on emissions and efficiency in plug-in hybrid vehicles. *Energy Conversion and Management: X*, 24:100776, 2024.

[137] European Commission. Commission regulation (eu) 2017/1151 of 1 june 2017 of the european parliament and of the council on type-approval of motor vehicles with respect to emissions from light passenger and commercial vehicles (euro 5 and euro 6) ... https://tinyurl.com/53ecjp7s, 2024, online documentation.

[138] Giuseppe Dipierro, Enrico Galvagno, Gianluca Mari, Federico Millo, Mauro Velardocchia, and Alessandro Perazzo. A reverse-engineering method for powertrain parameters characterization applied to a p2 plug-in hybrid electric vehicle with automatic transmission. volume 2020-June. SAE International, 6 2020.

[139] Federico Millo, Luciano Rolando, Luca Pulvirenti, and Giuseppe Di Pierro. A Methodology for the Reverse Engineering of the Energy Management Strategy of a Plug-In Hybrid Electric Vehicle for Virtual Test Rig Development. *SAE International Journal of Electrified Vehicles*, 11(1), 9 2021.

[140] A. Brahma, Yann Guezennec, and Giorgio Rizzoni. Optimal energy management in series hybrid electric vehicles. In *American Control Conference, 2000. Proceedings of the 2000*, volume 1, pages 60 – 64 vol.1, 10 2000.

[141] Ilya A Kulikov, Elena E Baulina, and Andrey I Filonov. Optimal control of a hybrid vehicle's powertrain minimizing pollutant emissions and fuel consumption. In *SAE Technical Paper 2014-09-30*, page 4, 2014.

[142] Federico Millo, Luciano Rolando, Fabio Mallamo, and Rocco Fuso. Development of an optimal strategy for the energy management of a range-extended electric vehicle with additional noise, vibration and harshness constraints. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 227(1):4–16, 2013.

[143] Chan-Chiao Lin, Jun-Mo Kang, J.W. Grizzle, and Huei Peng. Energy management strategy for a parallel hybrid electric truck. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 4, pages 2878–2883 vol.4, 2001.

[144] Domenico Bianchi, Luciano Rolando, Lorenzo Serrao, Simona Onori, Giorgio Rizzoni, Nazar Al-Khayat, Tung-Ming Hsieh, and Pengju Kang. Layered control strategies for hybrid electric vehicles based on optimal control. *International Journal of Electric and Hybrid Vehicles*, 3(2):191–217, 2011.

[145] Theo Hofman, Maarten Steinbuch, Roell Van Druten, and Alex Serrarens. Rule-based energy management strategies for hybrid vehicles. *International Journal of Electric and Hybrid Vehicles*, 1(1):71–94, 2007.

[146] Olle Sundstrom and Lino Guzzella. A generic dynamic programming matlab function. In *2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC)*, pages 1625–1630, 2009.

[147] K.F. Man, K.S. Tang, and S. Kwong. Genetic algorithms: concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics*, 43(5):519–534, 1996.

[148] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[149] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[150] M. Metghalchi and J. Keck. Burning velocities of mixtures of air with methanol, isooctane, and indolene at high pressure and temperature. *Combustion and Flame*, 48(2):191–210, 1982.

[151] Gerhard Woschni. A universally applicable equation for the instantaneous heat transfer coefficient in the internal combustion engine. *SAE transactions*, 76, 1967.

[152] I. I. Vibe. *Brennverlauf und Kreisprozeß von Verbrennungsmotoren*. VEB Verlag Technik, Berlin, 1970. Author often cited as I. I. Wiebe.

[153] James C. Keck, John B. Heywood, and Georg Noske. Early flame development and burning rates in spark ignition engines and their cyclic variability. In *SAE International Congress and Exposition*. SAE International, February 1987.

[154] F. Gouldin. An application of fractals to modeling premixed turbulent flames. *«!nav»Combustion and Flame«!/nav»*, 68(3):249–266, 1987.

[155] SEBASTIEN M. CANDEL and THIERRY J. POINSOT. Flame stretch and the balance equation for the flame area. *Combustion Science and Technology*, 70(1-3):1–15, 1990.

[156] Stéphane Richard and Denis Veynante. A 0-d flame wrinkling equation to describe the turbulent flame surface evolution in si engines. *Comptes Rendus Mécanique*, 343(3):219–231, 2015.

[157] Deng Hu, Hechun Wang, Chuanlei Yang, Binbin Wang, Baoyin Duan, Yinyan Wang, and Hucai Li. Construction of digital twin model of engine in-cylinder combustion based on data-driven. *Energy*, 293, 4 2024.

[158] Halit Yaşar, Gültekin Çağıl, Orhan Torkul, and Merve Şişci. Cylinder pressure prediction of an hcci engine using deep learning. *Chinese Journal of Mechanical Engineering (English Edition)*, 34, 12 2021.

[159] Özer Can, Tolga Baklacioglu, Erkan Özturk, and Onder Turan. Artificial neural networks modeling of combustion parameters for a diesel engine fueled with biodiesel fuel. *Energy*, 247, 5 2022.

[160] Moritz Sontheimer, Anshul Kumar Singh, Prateek Verma, Shuo Yan Chou, and Yu Lin Kuo. Lstm for modeling of cylinder pressure in hcci engines at different intake temperatures via time-series prediction. *Machines*, 11, 10 2023.

[161] Federico Ricci, Luca Petrucci, Francesco Mariani, and Carlo Nazareno Grimaldi. Investigation of a hybrid lstm + 1dcnn approach to predict in-cylinder pressure of internal combustion engines. *Information (Switzerland)*, 14, 9 2023.

[162] Hujun Peng, Jianxiang Li, Kai Deng, and Kay Hameyer. Machine learning-based control for fuel cell hybrid buses: From average load power prediction to energy management. *Vehicles*, 4(4):1365–1390, 2022.

[163] C. De Marino, G. Maiorana, P. Pallotti, S. Quinto, et al. The global small engine 3 and 4 cylinder turbo: The new fca's family of small high-tech gasoline engines. In *39th International Vienna Motor Symposium*, Vienna, Austria, April 2018.

[164] L. Bernard, A. Ferrari, D. Micelli, et al. Electro-hydraulic valve control with multiair technology. *MTZ Worldwide*, 2009.

[165] Toni Tahtouh, Mathieu André, Giuseppe Castellano, Luciano Rolando, and Federico Millo. A path toward a new generation of sustainable spark ignition engines: Experimental investigations on the synergic use of dual diluted combustion and renewable fuels. *Transportation Engineering*, 20:100317, 2025.

Appendix A

Additional Results

A.1 Modelling Application to Predictive Combustion Models

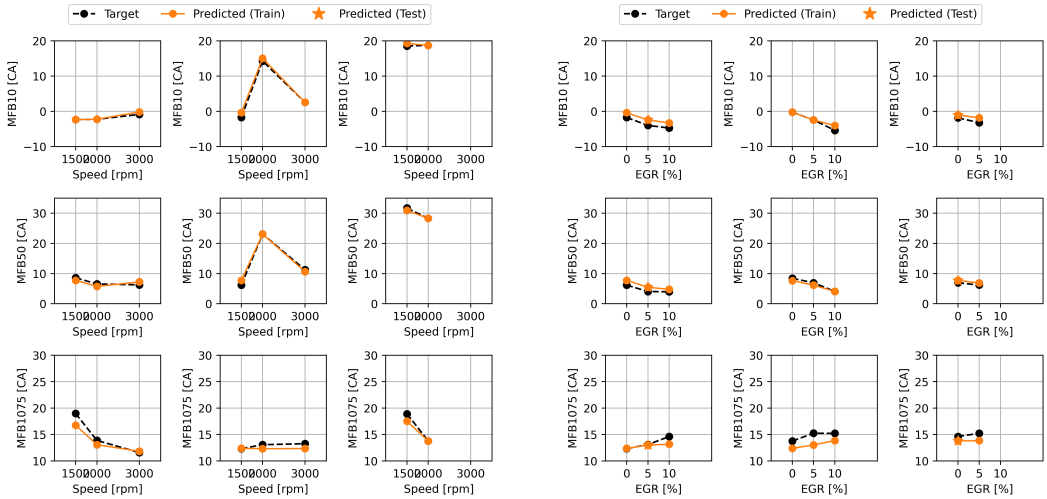
This appendix includes some additional plots used to further validate the scale 3 combustion model, both for the NN-Wiebe and for the Hybrid-RNN approach.

A.1.1 MLP for Wiebe Parameters Prediction

Figure A.1 reports the combustion metrics (MFB10, MFB50, MFB10-75) for (a) a speed–load sweep at EGR $= 0\%$ and (b) an EGR/λ sweep at 1500 rpm, 5.5 bar IMEP. Figure A.2 shows (a) burn–rate profiles and (b) the same combustion metrics for the EGR/λ sweep at 3000 rpm, 7 bar IMEP.

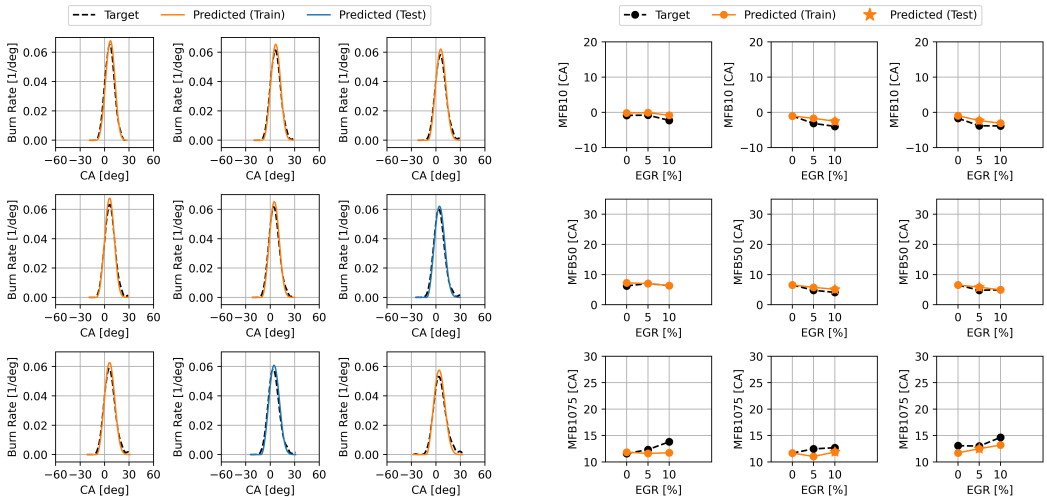
A.1.2 Hyrbid-LSTM for Burn Rate Prediction

Figure A.3 presents the regression plots of the combustion metrics for the training and validation sets. Figure A.4 reports the combustion metrics for EGR/λ sweeps at (a) 1500 rpm, 5.5 bar IMEP and (b) 3000 rpm, 7 bar IMEP.



(a) (b)

Fig. A.1 Combustion metric accuracy of the NN-Wiebe model. In (a) Speed-load sweep under stoichiometric conditions with EGR = 0%. In (b) EGR/λ sweep at 1500 rpm and 5.5 bar IMEP. Black markers are experimental targets; orange circles/lines are model predictions on training points; orange stars are predictions on test points.



(a) (b)

Fig. A.2 Burn rate profiles and combustion metric accuracy of the NN-Wiebe model. In (a) burn-rate profiles for the EGR/λ sweep at 3000 rpm and 7 bar IMEP. In (b) combustion metrics (MFB10, MFB50, and MFB10-75) for the same sweep. Black dashed lines/markers are experimental targets; orange circles/lines are model predictions on training points; orange stars are predictions on test points.

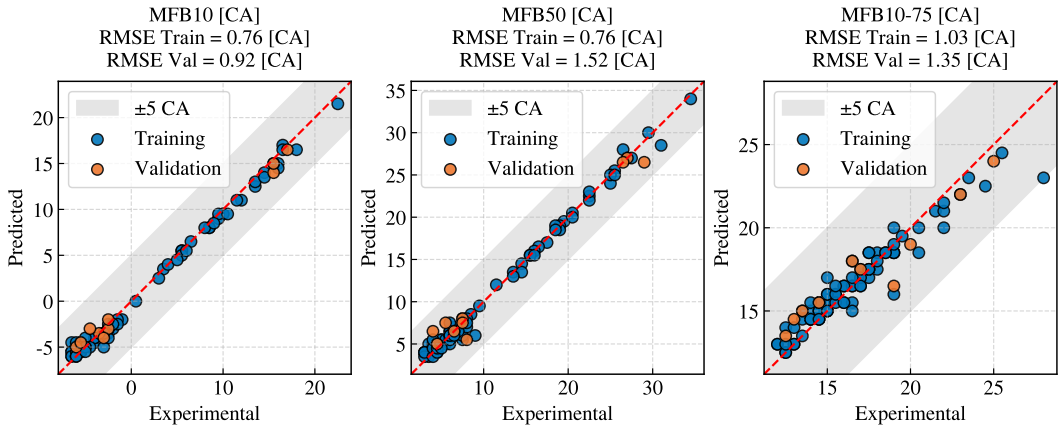
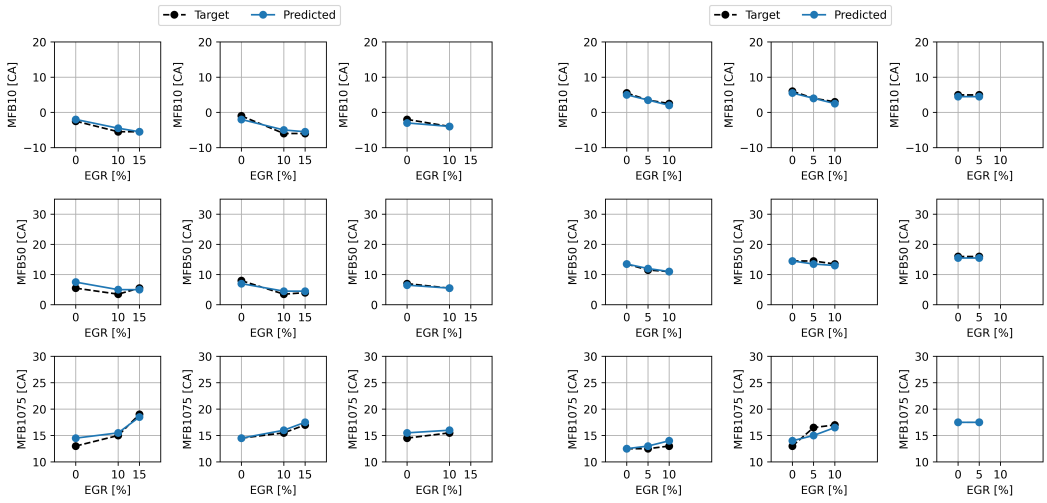


Fig. A.3 Regression of combustion metrics for the Hybrid-RNN model comparing training and validation sets. Scatter plots show predicted vs. experimental values for MFB10, MFB50, and MFB10-75. The red dashed line indicates the 1:1 reference; the shaded band highlights ±5 CA. Blue markers denote training points and orange markers denote validation points; panel headers report the corresponding RMSE values for each metric.



(a) (b)

Fig. A.4 Combustion metric accuracy of the Hybrid RNN model. In (a) EGR/λ sweep at 1500 rpm and 5.5 bar IMEP; in (b) EGR/λ sweep at 3000 rpm and 7 bar IMEP. Metrics shown are MFB10, MFB50, and MFB10-75. Black markers are experimental targets; blue circles/lines are model predictions.

Appendix B

List of Publications

Year 2025

1. Khan, M.A., Thatipamula, S., Tresca, L. et al. High-power lithium-ion battery characterization dataset for stochastic battery modeling. Sci Data 12, 1506 (2025). https://doi.org/10.1038/s41597-025-05725-y.

2. Tresca, L., Pulvirenti, L., Rolando, L., A Cutting-Edge Energy Management System for a Hybrid Electric Vehicle relying on Soft Actor–Critic Deep Reinforcement Learning, Transportation Engineering, 2025, https://doi.org/10.1016/j.treng.2025.100308.

3. Tresca, L., Pulvirenti, L., and Rolando, L., The Effect of the Initial Battery State of Charge on the Performance of a Soft Actor-Critic Energy Management System, SAE Technical Paper 2025-24-0104, 2025, https://doi.org/10.4271/2025-24-0104.

4. Tresca, L., Schmidt, C., Harrison, J., Rodrigues, F., Zardini, G., Gammelli, D., Pavone, M., Robo-taxi Fleet Coordination at Scale via Reinforcement Learning, 2025, ArXiv. https://arxiv.org/abs/2504.06125, under review by Transactions on Control of Network Systems.

Year 2024

1. Tresca, L., Pulvirenti, L., Rolando, L., Millo, F., Development of a deep Q-learning energy management system for a hybrid electric vehicle, Transportation Engineering, 2024, https://doi.org/10.1016/j.treng.2024.100241.

2. Rolando, L., Campanelli, N., Tresca, L., Pulvirenti, L. et al., "Development of a Soft-Actor Critic Reinforcement Learning Algorithm for the Energy Management of a Hybrid Electric Vehicle," SAE Int. J. Adv. & Curr. Prac. in Mobility 7(3):1140-1151, 2025, https://doi.org/10.4271/2024-37-0011.

Year 2023

1. Millo, F., Rolando L., Tresca, L., Pulvirenti, L., Development of a neural network-based energy management system for a plug-in hybrid electric vehicle, Transportation Engineering, 2023, https://doi.org/10.1016/j.treng.2022.100156.

2. Pulvirenti, L., Tresca, L., Rolando, L., Millo, F., Eco-Driving Optimization Based on Variable Grid Dynamic Programming and Vehicle Connectivity in a Real-World Scenario, Energies, 16(10), 4121, 2023, https://doi.org/10.3390/en16104121.

3. Galvagno, E., Rolando, L., Pulvirenti, L., Zerbato, L., Tresca, L., Hybrid Electric Vehicle Platoon Control and Optimal Energy Management System for Fuel Economy, Safety, and Dynamic Performance, In: Okada, M. (eds) Advances in Mechanism and Machine Science. IFToMM WC 2023. Mechanisms and Machine Science, vol 147. Springer, Cham. https://doi.org/10.1007/978-3-031-45705-0_95.