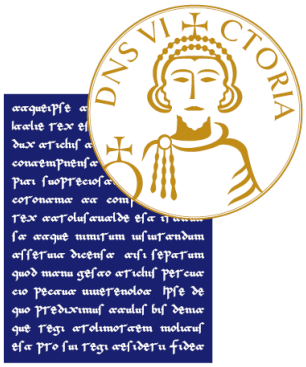
University of Sannio

Department of Engineering

Master's Degree in

Electronics Engineering for Automation and Sensing



OPTIMAL ELECTRIC VEHICLE BATTERY MANAGEMENT FOR VEHICLE-TO-GRID: MODEL PREDICTIVE CONTROL AND REINFORCEMENT LEARNING APPROACHES

Supervisor:
Prof. Carmela Bernardo

Co-Supervisor:
Dr. Antonio Pepiciello

Candidate:
Angelo Caravella
Student ID 389000016

Academic Year 2024–2025

Contents

Abstract in italian

L'adozione crescente dei **Veicoli Elettrici (EV)** in concomitanza con la sempre maggiore penetrazione di **Fonti di Energia Rinnovabile (RES)** intermittenti, presenta sfide significative alla **stabilità** e all'**efficienza della rete elettrica**. La tecnologia **Vehicle-to-Grid (V2G)** emerge come soluzione fondamentale, trasformando gli EV da carichi passivi a **risorse energetiche flessibili** capaci di fornire vari **servizi di rete**. Questa tesi affronta il complesso **problema di ottimizzazione multi-obiettivo** della gestione intelligente di carica e scarica degli EV, che intrinsecamente implica un equilibrio tra **benefici economici**, **esigenze di mobilità dell'utente**, **preservazione della salute della batteria** e **stabilità della rete** in condizioni stocastiche.

Di fronte alla complessa sfida di ottimizzare la ricarica dei veicoli elettrici (EV) in scenari Vehicle-to-Grid (V2G), un approccio che si limita a un singolo modello di controllo, come il Deep Q-Networks (DQN), risulterebbe inadeguato. La natura del problema, caratterizzata da molteplici obiettivi contrastanti (benefici economici, esigenze dell'utente, salute della batteria, stabilità della rete) e da una profonda incertezza; richiede un'analisi comparativa e rigorosa di un'ampia gamma di strategie di controllo. Per questo motivo, la ricerca si concentra sulla valutazione di un portafoglio diversificato di algoritmi, che include numerosi modelli di Deep Reinforcement Learning (DRL), approcci euristici e il Model Predictive Control (MPC). Questo metodo consente di mappare in modo completo il panorama delle soluzioni, identificando i punti di forza e di debolezza di ciascun approccio in relazione alle diverse sfaccettature del problema V2G.

In conclusione questa lavoro di tesi non si focalizza su un singolo modello, ma adotta un approccio comparativo su larga scala perché:

Non esiste una soluzione unica: La complessità del problema V2G rende improbabile che un solo algoritmo sia ottimale in tutte le condizioni.

Si ricercano i compromessi: L'obiettivo è comprendere i trade-off tra l'efficienza dei dati, la stabilità dell'addestramento, la robustezza all'incertezza e la complessità computazionale delle diverse famiglie di algoritmi.

La validazione è più rigorosa: Confrontare i modelli di DRL non solo tra loro ma anche con benchmark consolidati come le euristiche e l'MPC fornisce una misura molto più credibile del loro reale valore aggiunto.

Abstract

The growing adoption of **Electric Vehicles (EVs)**, combined with the increasing penetration of intermittent **Renewable Energy Sources (RES)**, presents significant challenges to the **stability** and **efficiency** of the power grid[1]. **Vehicle-to-Grid (V2G)** technology emerges as a key solution, transforming EVs from passive loads into **flexible energy resources** capable of providing various **grid services**. This thesis addresses the complex **multi-objective optimization problem** of smart EV charging and discharging, which requires balancing **economic benefits**, **user mobility needs**, **battery health preservation**, and **grid stability** under stochastic conditions.

Given the complexity of optimizing EV charging in V2G scenarios, relying on a single control model is insufficient. The nature of the problem, characterized by multiple conflicting objectives (economic benefits, user needs, battery health, grid stability) and profound uncertainty, demands a rigorous comparative analysis of a wide range of control strategies.

For this reason, the research focuses on evaluating a diverse portfolio of algorithms, including numerous Deep Reinforcement Learning (DRL) models, heuristic approaches, and Model Predictive Control (MPC). This method allows for a complete mapping of the solution landscape, identifying the strengths and weaknesses of each approach in relation to the different facets of the V2G problem.

In short, this thesis adopts a **broad-spectrum comparative approach** for several reasons:

No Single Solution: The complexity of the V2G problem makes it unlikely that a single algorithm can be optimal in all conditions.

Understanding Trade-offs: The goal is to understand the trade-offs between data efficiency, training stability, robustness to uncertainty, and the computational complexity of different algorithm families.

Rigorous Validation: Comparing DRL models not only against each other but also against established benchmarks like heuristics and MPC provides a more credible measure of their true value.

[1] 46.

List of Acronyms

Acronym	Description
Artificial Intelligence & Control	
A2C	Advantage Actor-Critic
AC	Actor-Critic
AI	Artificial Intelligence
AL-SAC	Augmented Lagrangian Soft Actor-Critic
ARS	Augmented Random Search
CL	Curriculum Learning
CMDP	Constrained Markov Decision Process
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Networks
DRL	Deep Reinforcement Learning
LQR	Linear Quadratic Regulator
LSTM	Long Short-Term Memory
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
MILP	Mixed-Integer Linear Program
MPC	Model Predictive Control
NN	Neural Network
PER	Prioritized Experience Replay
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SAC	Soft Actor-Critic
TD3	Twin-Delayed Deep Deterministic Policy Gradient
TQC	Truncated Quantile Critics
TRPO	Trust Region Policy Optimization
Electric Vehicles & Charging	
AFAP	As Fast As Possible (Heuristic)
ALAP	As Late As Possible (Heuristic)
CAFA	Charge As Fast As Possible
CALA	Charge As Late As Possible
CPO	Charge Point Operator
EV	Electric Vehicle
G2V	Grid-to-Vehicle
SCP	Scheduled Charging Power
SoC	State of Charge
SoH	State of Health
V2B	Vehicle-to-Building
V2G	Vehicle-to-Grid
V2H	Vehicle-to-Home
V2M	Vehicle-to-Microgrid
V2V	Vehicle-to-Vehicle
VPP	Virtual Power Plant

Acronym	Description
Power Grid & Energy Markets	
ACE	Area Control Error
ARR	Area Regulation Requirement
DER	Distributed Energy Resources
DR	Demand Response
RES	Renewable Energy Sources
Metrics & Technical Parameters	
DC	Constant Current (charging phase)
CV	Constant Voltage (charging phase)
DoD	Depth of Discharge
MSE	Mean Square Error
OU	Ornstein-Uhlenbeck (stochastic process)
RMSE	Root Mean Square Error

Chapter 1

Introduction

The global strategy for decarbonizing transport heavily relies on the shift toward electric mobility. This thesis investigates the complex challenges and opportunities arising from the large-scale integration of electric vehicles (EVs), as illustrated in Figure 1.1, into existing power grids.

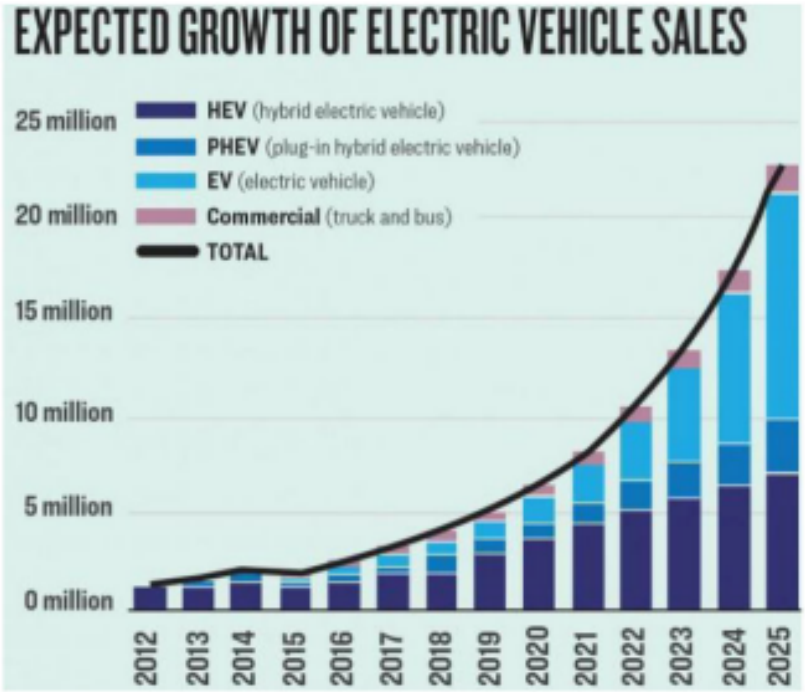


Figure 1.1: Expected growth of EV sales in the coming years (Image from: [1])

1.0.1 Background and Relevance of Electric Vehicles and Vehicle-to-Grid

> *"Electric car sales continue to break records globally, particularly in China and other emerging economies."*
>
> **International Energy Agency (IEA)** [2]

The rapidly expanding Electric Vehicle (EV) market is reshaping modern mobility, offering a path to reduced carbon emissions and enhanced energy efficiency[3]. This transition is fundamental to environmental sustainability, as it lessens dependence on fossil fuels, mitigates climate change by cutting greenhouse gas emissions, and improves urban air quality. However, integrating millions of EVs into the power system is a significant challenge, threatening to intensify peak demand, strain transmission and distribution networks, and cause technical issues like voltage irregularities or line losses[4].

This challenge raises a critical question: **can we transform this apparent liability into a foundational asset for grid stability?** The answer may be found in the Vehicle-to-Grid (V2G) paradigm. V2G reimagines EVs not as passive loads, but as mobile, flexible energy assets capable of bidirectional power exchange[5]. This potential is significant, considering EVs remain parked for approximately 96% of the day, providing a vast window for grid interaction[6]. Furthermore, the rapid responsiveness of EV batteries makes them ideal for providing ancillary services like frequency regulation[7]. A growing body of research, reviewed by Qiu et al.[8] and Xie[9], suggests that intelligent, bidirectional charging can offset the negative impacts of EV integration. The central proposition of this thesis is to test this hypothesis: that through advanced control methodologies, V2G can be proven not just theoretically sound, but practically indispensable, provided its economic and grid-stabilizing benefits demonstrably outweigh costs such as battery degradation and infrastructure investment.

Alongside V2G, other bidirectional power flow schemes, shown in Figure 1.2, have been proposed to enhance energy resilience:

[2] Global EV Outlook 2025 - Executive Summary
[3] 34.
[4] 34, 40.
[5] 2.
[6] 25.
[7] 2.
[8] 38.
[9] 52.

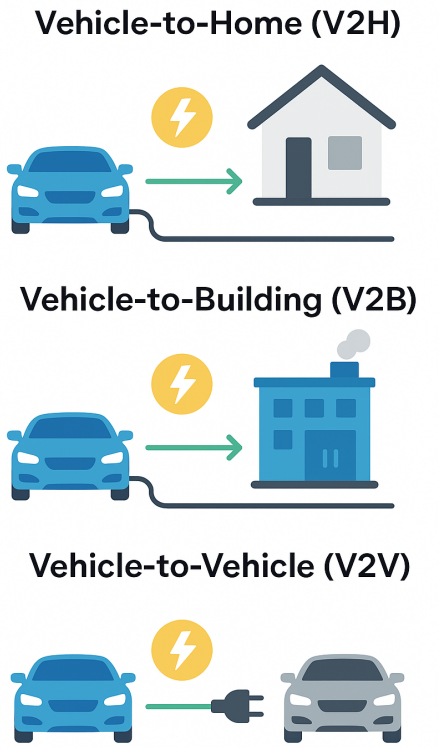


Figure 1.2: Other schemes of bidirectional power flow

1. **Vehicle-to-Home (V2H):** An EV powers a household during outages or high-cost periods, boosting domestic energy security.
2. **Vehicle-to-Building (V2B):** This concept is extended to commercial or industrial facilities, where EVs support load management and optimize energy consumption.
3. **Vehicle-to-Vehicle (V2V):** Direct power transfer between EVs provides a solution for emergency charging or resource sharing.

Collectively, these modalities underscore the versatility of EV batteries as distributed energy resources, advancing the transition to a more sustainable energy ecosystem.

1.0.2 Challenges in EV Integration into the Electricity Grid and the Role of Artificial Intelligence

Modern electricity systems face increasing strain from the integration of intermittent **Renewable Energy Sources (RESs)** like wind and solar. The inherent variability of RESs leads to significant power generation swings, creating supply-demand mismatches that fuel price volatility and complicate grid management. This continuous instability challenges the economic efficiency and reliability of

the grid, proving difficult for conventional control frameworks to manage[10].
The parallel rise of EV adoption and RES deployment has created an environment of unprecedented uncertainty and complexity. This situation makes traditional, rule-based controllers, designed for a more predictable and centralized grid, increasingly inadequate. **Is it possible, then, that a new control paradigm is needed?** The literature, as reviewed by NaXu et al.[11] and Feyijimi et al.[12], strongly suggests so, highlighting the limitations of legacy systems and framing the problem in a way that points toward data-driven methods. This shift signals a genuine paradigm change towards a *smart grid*, where adaptive, real-time, and autonomous operation is vital[13].
This has led to a surge in research focused on **Reinforcement Learning (RL)**, a paradigm that can, in theory, learn optimal control policies directly from environmental interaction without a perfect system model. While meta-heuristic algorithms have been explored, they often lack the real-time adaptability required for dynamic control[14]. The hypothesis to be tested is whether the theoretical promise of RL holds up against the engineering realities of the V2G problem. From this perspective, RL is not merely an optimization tool but an **enabling technology** for a more cognitive and robust energy infrastructure capable of navigating a decarbonized future. Within this domain, **Deep Reinforcement Learning (DRL)** has emerged as a particularly powerful approach, valued for its capacity to derive near-optimal strategies in dynamic and uncertain environments without relying on precise models or forecasts[15].

1.0.3 Objectives and Contributions of the Thesis

This thesis confronts the complex multi-objective optimization problem at the heart of Vehicle-to-Grid (V2G) systems. The overarching objective is to move beyond a purely theoretical analysis by actively developing, testing, and enhancing a high-fidelity simulation architecture. This platform serves as a digital twin to rigorously evaluate and compare advanced control strategies, balancing economic benefits, user mobility needs, battery health, and grid stability under realistic stochastic conditions.
More than a simple review of existing literature, this work focuses on the practical implementation and validation of a V2G simulation framework in Python. This tool is leveraged to demonstrate and explore novel perspectives for training intelligent agents. The main contributions are:

- **Enhancement of a V2G Simulation Architecture:** A key contribution is the systematic testing, validation, and enhancement of the **EV2Gym** simulation framework. This work solidifies its role as a robust platform for benchmark-

[10] 34, 30.
[11] 53.
[12] 1.
[13] 53.
[14] 44.
[15] 34.

ing control algorithms and includes the development of an interactive data application using **Streamlit** for results visualization and scenario analysis.

- **Development of a Battery Degradation Calibration Algorithm:** A novel algorithm for calibrating the battery degradation model is presented. This contribution ensures that the simulation's battery health predictions are grounded in realistic parameters, increasing the fidelity of the economic and physical assessments of V2G strategies.

- **Exploration of Novel Reinforcement Learning Perspectives:** The validated simulation environment is used to investigate and implement advanced training methodologies for RL agents. A key focus is placed on techniques like **adaptive reward shaping**, where the reward function dynamically evolves during training to guide the agent towards a more holistic and robust control policy.

- **Practical Implementation and Comparison of Advanced MPC Formulations:** The thesis details the development and implementation of multiple advanced model-based controllers. This includes an **explicit MPC**, a classic **implicit MPC**, and a novel **Adaptive Horizon Model Predictive Control (AHMPC)**, all formulated in PuLP. These are benchmarked against the theoretical offline optimal controller to rigorously analyze the trade-offs inherent in real-world, online deployment with limited future information.

1.0.4 Research Methodology

The research was guided by a central question: how can the economic profit of Vehicle-to-Grid (V2G) operations be maximized without imposing undue costs in terms of battery degradation or creating instability by overloading local grid infrastructure? To address this, a systematic methodology was adopted, rooted in the philosophical principle of **falsifiability** as articulated by Karl Popper[16]. The research is structured to formulate testable propositions that can be rigorously challenged by empirical evidence, where failure is as informative as success.
The initial phase involved an extensive literature review using Google Scholar to identify state-of-the-art V2G control strategies and their inherent trade-offs. This was followed by an empirical phase centered on the simulation framework detailed in Chapter 3. The significance of results from various control agents was continuously evaluated through a multi-faceted validation process, including comparative analysis against published benchmarks, heuristic evaluation based on acquired expertise, and grounding empirical findings in the foundational knowledge from the literature review.
To further systematize the literature review, a quantitative analysis was performed on the collected papers using a custom Python script. This analysis produced a key visualization: a weighted word cloud (Figure 1.3), was generated from titles and abstracts, with word size corresponding to its **PageRank** score within a citation graph.

[16] 33.

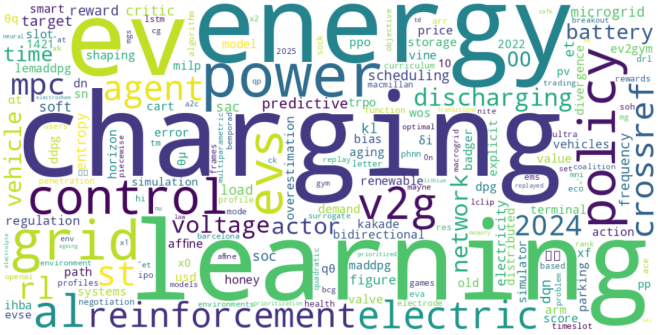


Figure 1.3: Weighted word cloud generated from the literature review, highlighting key research themes. The size of each word corresponds to its frequency and importance, derived from the PageRank analysis.

This visualization confirms the research focus, with primary terms like **Energy**, **Power**, **Grid**, and **EVs** setting the context. The equal prominence of **Learning** and **Control**, alongside strong secondary terms like **Reinforcement**, **Agent**, and **Policy**, firmly anchors the methodology in Reinforcement Learning. The inclusion of **MPC** (Model Predictive Control) highlights the central dialogue between model-free and model-based approaches explored in this thesis.

1.0.5 Thesis Structure

The remainder of this thesis is organized as follows:

- **Chapter 2: Overview of Optimal Management of EV Charging and Discharging** provides foundational knowledge on V2G technology, the complex multi-objective nature of EV charging optimization, and presents a comprehensive review of state-of-the-art research approaches.
- **Chapter 3: The V2G Simulation Framework: A Digital Twin for V2G Research** details the architecture and core models of the simulation environment. This chapter describes the enhancements made to the framework, establishing it as the central experimental platform for implementing and evaluating the control agents analyzed in this work.
- **Chapter 4: Experimental Campaign and Results Analysis** presents the main results from the comparative analysis of the different control strategies (DRL, MPCs, heuristics). It analyzes the performance of novel training techniques, discusses the implications of the findings, and provides a detailed **sensitivity analysis** on the most critical system parameters to assess the robustness of the conclusions.
- **Bibliography** lists all cited references.

Chapter 2

State of the Art in Optimal V2G Management

> *"The green transition is the most ambitious industrial transformation ever. The region of the world that develops clean technologies first will come out on top — and I want it to be Europe."*
>
> — Ursula von der Leyen

2.1 The V2G Imperative: A Foundation of Europe's Green Transition

Europe finds itself at the confluence of two unprecedented and deeply interlinked transformations reshaping its technological, economic, and societal landscape: the large-scale electrification of transport and a comprehensive restructuring of its energy systems. These parallel transitions involve a radical shift from internal combustion engines to battery electric vehicles and a simultaneous integration of renewable generation, grid modernization, and the deployment of advanced storage and demand-side management solutions.
These transformations are not merely aspirational targets but constitute binding legal obligations established under the **European Green Deal** and the detailed **"Fit for 55"** legislative package. These frameworks translate climate ambitions into enforceable measures aimed at reducing net greenhouse gas emissions by 55% by 2030[1]. This policy architecture necessitates the rapid phase-out of fossil-fuelled vehicles alongside a dramatic expansion of renewable energy capacity, a goal further reinforced by the revised **Renewable Energy Directive (RED III)**, as detailed in Figure 2.1.

[1] 12.

	RED II (2018)	RED III (2023)
RENEWABLE ENERGY TARGET	32% by 2030	42.5% by 2030 (45% aspirational target)
TRANSPORT SECTOR TARGET	14% renewable energy	29% renewable energy, 14.5% GHG intensity reduction in transport
GHG SAVINGS THRESHOLD FOR BIOFUELS	50–65% depending on installation date	70% (existing), 80% (new installations)
MASS BALANCE TRACEABILITY	Encouraged	Mandatory
ENFORCEABILITY	Partially voluntary or indicative	Legally binding and auditable
CHAIN OF CUSTODY SYSTEMS	Not required	Required across the entire value chain
ALIGNMENT WITH OTHER EU LEGISLATION	Limited	Integrated with ETS, CBAM, and EUDR frameworks

Figure 2.1: Comparison of key targets between the Renewable Energy Directive II (RED II, 2018) and the revised RED III (2023). The figure highlights the significantly increased ambition in RED III, including a higher overall renewable energy target, a more aggressive goal for the transport sector, and stricter requirements for traceability and legal enforceability. This escalation in policy ambition underscores the critical need for flexibility solutions like V2G to manage the grid and integrate higher shares of renewables.

Electric Vehicles (EVs) occupy a central position in this transition, simultaneously driving decarbonisation efforts while presenting complex challenges for grid stability. The initial response to mass EV adoption was characterised by concern within the power sector, viewing millions of new EVs as vast, correlated loads threatening to overwhelm distribution networks. This perspective has undergone a fundamental reassessment. EVs are now recognised not as burdens to be managed, but as essential, flexible assets for achieving Europe's energy objectives.
This conceptual shift finds its most concrete expression in **Vehicle-to-Grid (V2G)** technology, which fundamentally reimagines the role of electric vehicles within the energy system.
V2G technology transforms previously passive, unidirectional energy consumers into active, distributed, and intelligent grid resources. The underlying opportunity is substantial: private vehicles spend approximately 96% of their operational lifetime parked[2], representing an enormous, geographically distributed, and currently underutilised repository of mobile energy storage.
The transformative potential of V2G becomes apparent when individual vehicles are coordinated through centrally managed aggregation. While a single EV's contribution is modest, an orchestrated fleet can function as a unified **Virtual Power Plant (VPP)**. These software-defined power plants aggregate the collective capacity of numerous distributed energy resources, delivering grid services at scales comparable to conventional generation facilities. The rapid response characteristics of contemporary battery inverters, operating at millisecond timescales, enable these aggregated fleets to provide a comprehensive range of critical grid services. This capability is a prerequisite for maintaining stability in grids increasingly dependent on variable wind and solar generation, thereby enabling the technical

[2] 25.

and economic viability of the EU's ambitious renewable energy targets[3].

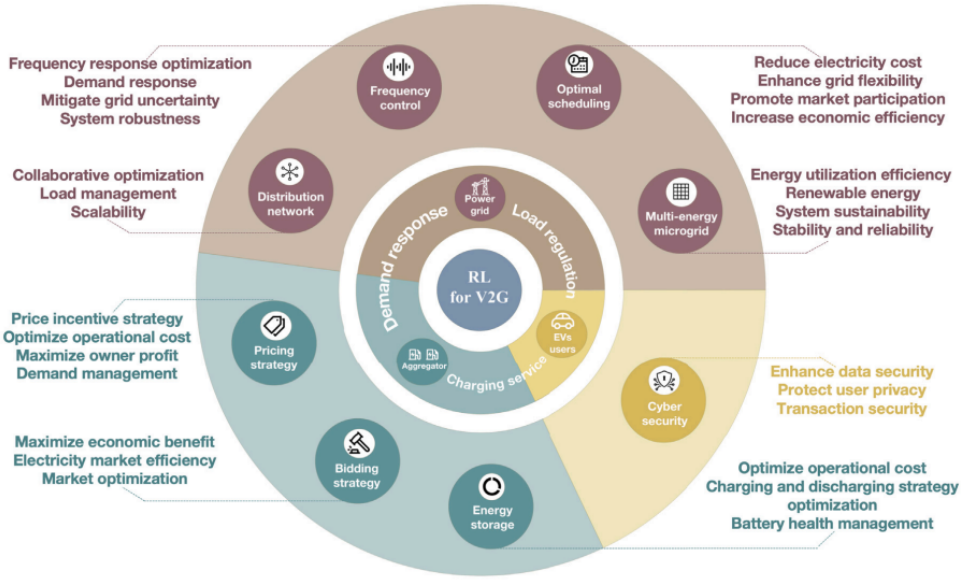


Figure 2.2: Reinforcement learning (RL) applications in V2G from the perspective of participating entities. This diagram illustrates the multifaceted roles an EV fleet can play, orchestrated by an aggregator using RL. Key operations include grid-stabilizing services (frequency control, demand response), economic optimization (bidding and pricing strategies), and user-centric management (energy storage, battery health). The objectives range from reducing electricity costs and enhancing grid flexibility to ensuring data privacy and transaction security, showcasing the complexity of optimizing V2G operations.

The grid services enabled by V2G technology, many of which are illustrated in Figure 2.2, form the technical foundation for the smart, resilient, and decarbonised electricity system required for Europe's energy future:

Frequency Regulation: Grid stability depends on maintaining a precise equilibrium between electricity supply and demand, manifested as a stable grid frequency (50 Hz in Europe). Deviations signal imbalances that can trigger cascading failures. V2G fleets, with their rapid-response inverters, can participate in ancillary service markets like Frequency Containment Reserve (FCR) and automatic Frequency Restoration Reserve (aFRR), injecting or absorbing power within seconds to counteract deviations and prevent blackouts[4].

Demand Response and Peak Shaving: By intelligently shifting charging to off-peak periods and strategically discharging during peak demand, V2G systems flatten daily load profiles. This directly mitigates the "duck curve" phenomenon (Figure 2.3) associated with high solar penetration. Such load management reduces reliance on expensive, carbon-intensive "peaker" plants and can defer or

[3] 46.
[4] 2, 51.

eliminate costly grid infrastructure upgrades[5].

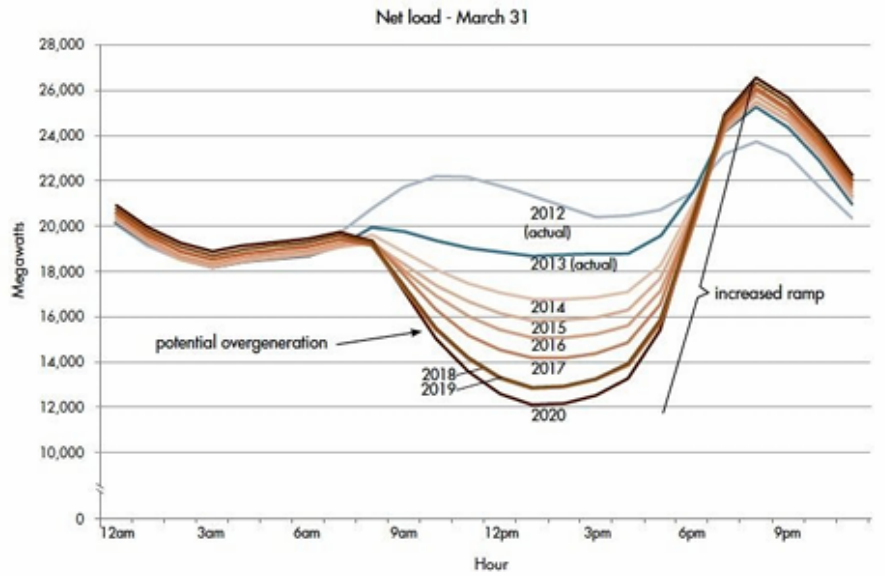


Figure 2.3: The "Duck Curve" illustrating the evolution of net load on the California grid. Net load is the total electricity demand minus variable renewable energy generation. The deepening "belly" of the duck during midday reflects overgeneration from solar power, while the steep "neck" in the evening represents a rapid increase in demand as solar generation fades. This steep ramp poses a significant challenge for grid operators. V2G helps mitigate this by absorbing excess energy during the day (charging) and discharging it during the evening ramp to "flatten the curve".[7]

Renewable Energy Integration: The most strategically significant contribution of V2G is addressing renewable energy intermittency. V2G fleets act as large-scale energy buffers, absorbing excess solar and wind generation that would otherwise be curtailed. This stored energy is then released during periods of low generation. This mechanism directly increases the utilisation of renewable resources, supporting the integration objectives of RED III and enhancing overall system efficiency.

Economic and Market Optimization: Beyond grid stability, V2G enables sophisticated market participation. As shown in Figure 2.2, aggregators can deploy intelligent bidding and pricing strategies. Reinforcement learning algorithms can learn optimal policies for participating in day-ahead and intraday energy markets, maximizing revenue for both the aggregator and EV owners by capitalizing on price volatility.

User-Centric Management and Security: For V2G to succeed, it must align with user needs and address concerns. This includes optimizing charging and discharging cycles to minimize **battery degradation**, thus preserving vehicle lifespan. Furthermore, as V2G involves financial transactions and data exchange, robust **cyber security** is paramount to protect user privacy and ensure transaction

[5]34, 39.

security, another key domain for intelligent management systems.
This vision is being actively incorporated into European legal frameworks. The transformative **Alternative Fuels Infrastructure Regulation (AFIR, EU 2023/1804)** now mandates that new public charging infrastructure incorporate smart and bidirectional capabilities. This is supported by technical standards like **ISO 15118-20**, which specifies protocols for a "Vehicle-to-Grid Communication Interface" (V2GCI). With mandatory implementation scheduled for 2027, the necessary infrastructure is being systematically established, supported by pilot initiatives like the **'SCALE'** and **'V2G Balearic Islands'** projects.
Despite this progress, substantial obstacles to widespread V2G deployment persist:

- **Market and Economic Barriers:** A coherent, pan-European framework for compensating EV owners for grid services is still undeveloped. Accessing existing value streams is complex, and issues like the **"double taxation"** of electricity—taxing energy during both charging and discharging—create significant economic disincentives.
- **Regulatory and Grid Access Challenges:** The qualification of EV fleets as flexibility resources varies across national markets. Standardised procedures for grid interconnection, aggregator certification, and secure data exchange are needed to reduce fragmentation and simplify commercial deployment.
- **Technical and Consumer Adoption Barriers:** Consumer concerns regarding accelerated **battery degradation** and its impact on vehicle warranties are primary obstacles. Furthermore, much of the current EV and charging infrastructure lacks bidirectional capability, though this is changing with new vehicle platforms and standards.

The fundamental challenge addressed by this thesis extends beyond simply enabling V2G technology to encompass its *intelligent orchestration*. This requires developing control strategies sophisticated enough to operate within emerging regulatory frameworks, navigate economic uncertainties, accommodate diverse user preferences, and overcome technical constraints. The objective is to unlock the substantial potential of EVs as fundamental components of Europe's energy transition while ensuring system reliability, economic viability, and user acceptance.

2.2 The Optimizer's Trilemma: Navigating a Stochastic World

> *"Uncertainty quantification provides a systematic way to assess the credibility of computational predictions."*

While the potential of V2G technology is substantial, the management of distributed vehicular assets presents a complex control challenge. Economic viability drives aggregator decisions, yet a narrow focus on profitability alone proves insufficient for sustainable operations. Effective V2G management requires balancing three competing objectives that frequently conflict with one another. This challenge can be framed as the "V2G Optimizer's Trilemma": the concurrent pursuit of **economic profitability**, preservation of **battery longevity**, and maintenance of **user convenience**. Rather than representing a straightforward, static trade-off, this constitutes a dynamic, multi-objective optimisation challenge characterised by **stochasticity** and **uncertainty** arising from multiple, interconnected sources[8]:
Market Volatility: Wholesale electricity prices exhibit significant variability driven by unpredictable supply variations (such as sudden reductions in wind generation capacity) and demand fluctuations (including heat-driven increases in cooling demand). Effective control systems must respond to these price signals dynamically and in real-time.
Renewable Intermittency: Co-located solar and wind generation exhibit inherently variable output patterns with limited predictability. Controllers must coordinate EV fleet operations to capture available generation during surplus periods without compromising other operational objectives.
Human Behaviour: Perhaps the most challenging uncertainty source involves EV owner patterns. Arrival times, departure schedules, and required state of charge (SoC) at departure lack deterministic characteristics. Emergency departures or unexpected schedule changes represent hard, non-negotiable constraints that intelligent systems must accommodate to preserve user trust and satisfaction.
This dynamic, uncertain, and multifaceted operational environment renders static, rule-based control approaches (such as "charge when price falls below threshold X, discharge when exceeding threshold Y") inadequate and brittle. More sophisticated and adaptive methodologies are required,approaches capable of learning from operational experience and making optimal decisions under conditions of significant uncertainty. Reinforcement Learning excels in precisely this domain, providing a framework for developing control policies that demonstrate robustness, adaptability, and scalability.

2.2.1 Sources for Energy Price Data

Access to reliable, real-time, and historical market data remains crucial for both control agent training in simulation environments and real-world deployment. Key public sources for European market data include:
ENTSO-E Transparency Platform: The European Network of Transmission System Operators for Electricity maintains a mandatory, open-access platform serving as a comprehensive repository of pan-European electricity market data. This includes harmonised day-ahead prices, load forecasts, and generation data, serving

[8] 49.

as the primary source for academic research through both web portal access and free RESTful API services.
National Transmission System Operators (TSOs): Many national TSOs (including Terna in Italy, National Grid in the UK, and RTE in France) publish detailed market data covering real-time frequency and imbalance prices for their respective jurisdictions.
Power Exchanges: Exchanges such as **EPEX SPOT** and **Nord Pool** constitute actual trading venues. While they represent direct price data sources, comprehensive real-time access typically requires commercial subscription services.

2.2.2 Buying vs. Selling: The Critical Retail-Wholesale Spread

> *"Price spreads, or marketing margins, are the difference between prices at different stages of the supply chain. The wholesale-to-retail spread is the difference between the wholesale price and the retail price."*

— Sebastien Pouliot & Lee L. Schulz, *Measuring Price Spreads in Red Meat* (2016)

A critical yet frequently overlooked aspect of V2G economics involves the distinction between EV owner charging costs and aggregator grid sales revenue.

- **Selling Price (V2G Revenue):** When EVs provide energy to the grid, revenue calculation bases on **wholesale prices** (such as day-ahead spot prices). These prices reflect pure marginal energy costs at specific times.
- **Buying Price (Charging Cost):** End consumer EV charging costs reflect **retail prices**, significantly exceeding wholesale prices due to numerous non-energy components, termed "non-commodity costs":
  - Base wholesale energy costs
  - **Grid Tariffs:** Charges for high-voltage transmission and low-voltage distribution network usage
  - **Taxes and Levies:** National or regional taxation including VAT and environmental levies applied to electricity consumption
  - **Supplier Margin:** Retail energy provider profit margins

This substantial gap between retail purchasing prices and wholesale selling prices constitutes the "retail-wholesale spread," creating the primary opportunity for profitable energy arbitrage. Successful control strategies must account for these price differentials to enable economically rational decision-making.
A further perspective on this issue is provided by Parisio et al.[9] (2014), who

[9] 35.

develop a model predictive control (MPC) framework for microgrid operation. Their formulation explicitly considers the decision of when to buy from or sell to the utility grid, under time-varying spot prices and operational constraints. Importantly, the model prevents physically and economically unrealistic behaviors such as simultaneous buying and selling, and accounts for the real cost of storage and generation. This reinforces the notion that optimal energy management strategies must capture the full set of economic signals—including retail charges, network tariffs, and non-commodity costs,rather than relying solely on wholesale price arbitrage. In the V2G context, this highlights the need for predictive, multi-constraint optimization frameworks capable of managing battery limitations, retail–wholesale spreads, and market participation simultaneously in order to ensure profitability.

2.3 Modelling the V2G Ecosystem

Before examining control algorithms, establishing clear, high-fidelity models of system core components becomes essential: the electric vehicle as a controllable cyber-physical asset, and the operational environment or "scenario." The interaction between these elements defines V2G optimisation task boundaries and objectives.

2.3.1 The Grid-Interactive EV as a Controllable Asset

From a power grid perspective, electric vehicles represent sophisticated mobile energy storage devices. For V2G applications, EVs can be characterised through several key state variables and parameters:
The Battery: The core grid asset component, defined by **nominal energy capacity** (in kWh), current **State of Charge (SoC)**, and **State of Health (SoH)** representing degradation over time. Operation is constrained by **power limits** (in kW) dictating maximum charge or discharge rates, and charging/discharging **efficiencies** accounting for energy losses.
The On-Board Charger (OBC): For AC charging applications, the OBC converts grid alternating current to battery direct current. Power rating often constitutes the primary bottleneck for both charging and V2G power output.
Communication Interface: V2G participation requires vehicle-charging station (EVSE) communication capabilities. This is governed by standards including **ISO 15118** and protocols such as the **Open Charge Point Protocol (OCPP)**, enabling secure information exchange required for smart and bidirectional power flow operations.
Combined with vehicle availability patterns—arrival and departure times plus user energy requirements,these characteristics transform EVs from simple loads into fully dispatchable grid resources.

2.4 A New Paradigm for Control: Reinforcement Learning - Based on the work of Sutton & Barto

> *"Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal."*
>
> — Richard S. Sutton & Andrew G. Barto, *Reinforcement Learning: An Introduction* (2018)[10]

To address the complexities of uncertainty, multi-objective trade-offs, and dynamic systems, this work employs Reinforcement Learning (RL), a machine learning paradigm that learns optimal sequential decision-making policies through trial-and-error interaction with an environment. Unlike traditional optimal control methods, which depend on an explicit and accurate model of the environment's dynamics, RL agents learn directly from the outcomes of their actions. This model-free approach provides significant robustness in the face of uncertainty and unmodeled dynamics.

2.5 The Reinforcement Learning Problem

The problem of reinforcement learning is formalized as the interaction between a learning **agent** and its **environment**. This interaction unfolds over a sequence of discrete time steps, $t = 0, 1, 2, \ldots$.

2.5.1 The Agent-Environment Interface

At each time step t, the agent receives a representation of the environment's **state**, $S_t \in \mathcal{S}$, and on that basis selects an **action**, $A_t \in \mathcal{A}(S_t)$. One time step later, as a consequence of its action, the agent receives a numerical **reward**, $R_{t+1} \in \mathcal{R}$, and finds itself in a new state, S_{t+1}. This interaction loop forms the foundational framework of the RL problem.

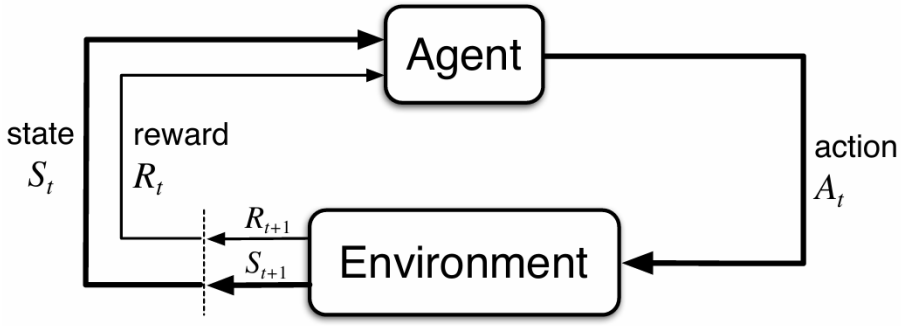


Figure 2.4: The agent-environment interaction loop in reinforcement learning[11].

[10] 45.

2.5.2 Goals, Rewards, and Returns

The agent's objective is formalized by the **reward hypothesis**: that all goals and purposes can be framed as the maximization of the expected cumulative reward. The agent's goal is not to maximize the immediate reward, R_{t+1}, but the cumulative reward in the long run. This cumulative reward is known as the **return**, denoted G_t.
For *episodic tasks* that terminate, the return is the finite sum of future rewards. For *continuing tasks* that do not terminate, the return is defined as the discounted sum of future rewards:

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

where $\gamma \in [0, 1)$ is the **discount factor**. It determines the present value of future rewards, ensuring the infinite sum is finite and balancing immediate gratification against long-term gains. A value of $\gamma = 0$ results in a myopic agent concerned only with maximizing immediate rewards.

2.6 The Language of Learning: Markov Decision Processes

The mathematical foundation of Reinforcement Learning (RL) is the **Markov Decision Process (MDP)**. An MDP provides a formal framework for modeling decision-making in stochastic environments where outcomes are partly random and partly under the control of a decision-maker.
An MDP is formally defined as a tuple $\langle S, A, P, R \rangle$, where:

- S is a finite set of states.
- A is a finite set of actions.
- P is the state transition probability function, $P(s'|s, a)$, which represents the probability of transitioning to state s' from state s after taking action a.
- R is the reward function, $R(s, a, s')$, which is the immediate reward received after transitioning from state s to state s' as a result of action a.

A key assumption in an MDP is that the environment is fully observable, meaning the agent knows its current state with certainty. However, in many real-world scenarios, the agent may not have complete information about its state. This is where the **Partially Observable Markov Decision Process (POMDP)** comes into play.
A POMDP extends the MDP framework to situations where the agent's observations of the environment are incomplete or noisy. A POMDP is represented by a tuple $\langle S, A, P, R, \Omega, O \rangle$, which includes all the elements of an MDP plus:

- Ω is a finite set of observations the agent can receive from the environment.

- O is the observation probability function, $O(o|s', a)$, which is the probability of observing o after transitioning to state s' having taken action a.

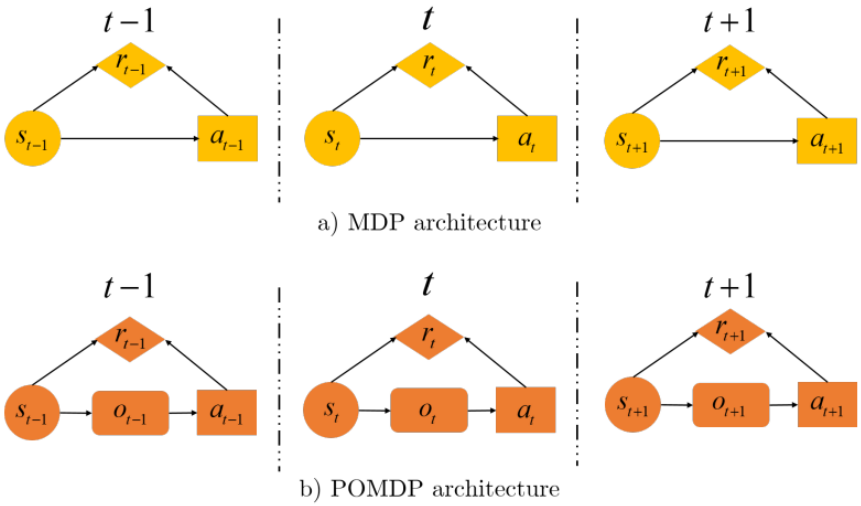


Figure 2.5: Differences between POMDP and MDP[12].

The fundamental difference between an MDP and a POMDP lies in the agent's perception of the environment's state. In an MDP, the agent directly observes the state s. In a POMDP, the agent receives an observation o and must infer a belief, which is a probability distribution over the possible states, to make a decision. As discussed in Sadeghi's work, this partial observability in POMDPs introduces a significant layer of complexity because the agent must act based on a belief state rather than a certain state.[13]
In our current discussion, we will focus on the MDP framework, assuming that the state of the environment is fully observable to the agent.

2.6.1 The Markov Property

The future is independent of the past given the present. A state signal S_t is said to have the **Markov Property** if the environment's response at time $t+1$ depends only on the state and action at time t. The probability of transitioning to state s' and receiving reward r is independent of all previous states and actions:

$$p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{2.2}$$

This property is fundamental as it allows decisions to be made based solely on the current state, without needing the complete history of interaction.

[13] 39.

2.6.2 Policies and Value Functions

The agent's learning objective is to find a good **policy**, $\pi(a|s)$, which is a mapping from states to probabilities of selecting each possible action. The goodness of a policy is assessed by its **value functions**.

- The **state-value function**, $v_\pi(s)$, is the expected return starting from state s and following policy π thereafter:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] \tag{2.3}$$

- The **action-value function**, $q_\pi(s, a)$, is the expected return starting from state s, taking action a, and thereafter following policy π:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{2.4}$$

2.7 The Bellman Equations

The Bellman equations provide a recursive decomposition that is foundational to solving MDPs. They express the value of a state in terms of the values of its successor states.

2.7.1 The Bellman Expectation Equation

For a given policy π, the state-value function must satisfy a self-consistency condition. The value of a state equals the expected immediate reward plus the discounted expected value of the next state:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.5}$$

This is the **Bellman expectation equation** for v_π. It forms the basis for policy evaluation algorithms.

2.7.2 The Bellman Optimality Equation

The ultimate goal is to find an **optimal policy**, π_*, which is a policy that achieves a higher or equal expected return than all other policies from all states. All optimal policies share the same optimal value functions, $v_*(s)$ and $q_*(s, a)$. The optimal value function for a state is the maximum expected return achievable from that state:

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{2.6}$$

This is the **Bellman optimality equation**. Solving it means finding the optimal policy.

2.7.3 Generalized Policy Iteration (GPI)

Most RL algorithms can be understood within the framework of **Generalized Policy Iteration (GPI)**, as described in Chapter 4 of[14]. GPI refers to the general idea of letting two interacting processes, policy evaluation and policy improvement, work towards a common optimal solution.

- **Policy Evaluation**: Given a policy π, compute its value function v_π. This step aims to make the value function consistent with the current policy.
- **Policy Improvement**: Given a value function v, improve the policy by making it greedy with respect to v. For a given state s, the new policy will select the action a that maximizes $q_\pi(s, a)$.

These two processes compete in the short term (improving the policy makes the value function inaccurate) but cooperate in the long term to converge to the optimal policy and optimal value function. **Dynamic Programming (DP)** methods like policy iteration and value iteration are classic examples of GPI, assuming a perfect model of the environment.

2.8 Learning from Experience: MC and TD Methods

When a model of the environment is not available, we must learn from sampled experience. Chapters 5 and 6 of[15] introduce two primary model-free approaches.

2.8.1 Monte Carlo (MC) Methods

MC methods learn value functions by averaging the returns from sample episodes.

- **Principle**: An update to $V(S_t)$ is made only at the end of an episode.
- **Update Target**: The target for the update is the actual, complete return G_t.
- **Update Rule** (Constant-α): $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$
- **Properties**: MC methods are unbiased but can have high variance and are only applicable to episodic tasks. They do not *bootstrap* [16].

2.8.2 Temporal-Difference (TD) Learning

TD learning is a central and novel idea in RL, combining ideas from both MC and DP.

- **Principle**: TD methods update the value estimate for a state based on the observed reward and the estimated value of the successor state. They learn from incomplete episodes.

[14] 45.
[15] 45.
[16] It means that MC updates only from complete sampled returns.

- **Update Target**: The target is an estimate of the return, called the TD Target: $R_{t+1} + \gamma V(S_{t+1})$.
- **Update Rule** (TD(0)): $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
- **Properties**: TD methods *bootstrap*:they update a guess from a guess. This introduces bias but often leads to lower variance and faster learning. They are naturally implemented in an online, fully incremental fashion.

2.9 Actor-Critic Architectures

The **Actor-Critic** architecture provides a powerful and widely adopted method for solving RL problems, particularly in continuous action spaces. It explicitly represents both the policy and the value function using two distinct function approximators (e.g., neural networks).

- **The Critic**: Learns a value function (e.g., $v_\pi(s)$ or $q_\pi(s, a)$). Its role is to evaluate the actor's decisions by computing the TD error:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2.7}$$

- **The Actor**: Represents the policy, $\pi_\theta(a|s)$, parameterized by θ. It receives the current state as input and outputs an action. It uses the TD error from the critic as a learning signal to update its parameters via policy gradient methods, improving its strategy over time.

This separation of concerns allows for direct policy optimization in continuous or large action spaces while leveraging the stable learning dynamics of TD-based value estimation.

2.10 Reward Engineering: Shaping Agent Behavior

The design of an effective reward function is arguably the most critical aspect of any Reinforcement Learning (RL) system. It serves as the primary communication channel through which designers convey desired behaviors and objectives to a learning agent. A poorly designed reward function can lead to agents learning unintended, suboptimal, or even harmful behaviors,despite successfully maximizing their assigned objective. Consequently, reward engineering has emerged as a fundamental and increasingly sophisticated discipline within modern RL, proving indispensable for the successful application of algorithms in complex, real-world scenarios[17]. The Vehicle-to-Grid (V2G) challenge, with its inherent multi-objective nature,encompassing profit maximization, assurance of user satisfaction, preservation of battery health, and maintenance of grid stability—presents particularly stringent demands on reward function design. This section investigates several advanced techniques for effectively guiding agent learning in such intricate environments.

[17] 20.

2.10.1 Potential-Based Reward Shaping (PBRS)

Potential-Based Reward Shaping (PBRS) stands as one of the most theoretically grounded and widely adopted methods for augmenting an environment's intrinsic reward signal. The core idea behind PBRS is to supplement the original reward, $R(s, a, s')$, received by an agent for transitioning from state s to state s' via action a, with an additional shaping term, $F(s, s')$. The resulting shaped reward, R', is defined as:

$$R'(s, a, s') = R(s, a, s') + F(s, s')$$

The crucial characteristic of PBRS lies in the specific construction of the shaping term. It is derived from the difference in value of an arbitrary potential function, $\Phi : \mathcal{S} \to \mathbb{R}$, evaluated at the successive states s and s':

$$F(s, s') = \gamma\Phi(s') - \Phi(s)$$

where $\gamma \in [0, 1)$ is the discount factor of the Markov Decision Process (MDP). The potential function $\Phi(s)$ assigns a scalar value to each state, intuitively representing how "good" or "desirable" that state is.

2.10.2 Theoretical Foundation and Policy Invariance

The seminal work by Ng et al.[18] established the theoretical robustness of PBRS by proving a critical property: **policy invariance**.
This property guarantees that adding a potential-based shaping reward to an MDP does not alter its set of optimal policies.
In other words, **any policy that is optimal for the shaped reward function R' will also be optimal for the original reward function R, and vice versa**. This is a profound result because it ensures that while PBRS can significantly accelerate learning by providing denser and more informative feedback, it will not mislead the agent into converging on a suboptimal policy with respect to the original task.
The proof of policy invariance hinges on the observation that the shaping term $F(s, s')$ can be absorbed into the value function. Specifically, if $V^*(s)$ and $Q^*(s, a)$ are the optimal value and Q-functions for the original MDP with reward R, then the optimal value and Q-functions for the shaped MDP with reward R' are given by:

$$V^{*'}(s) = V^*(s) + \Phi(s)$$
$$Q^{*'}(s, a) = Q^*(s, a) + \Phi(s)$$

Since the $\Phi(s)$ term is added uniformly to all Q-values for a given state s, the action that maximizes $Q^{*'}(s, a)$ will be the same action that maximizes $Q^*(s, a)$. This preserves the optimal policy:

$$\pi^{*'}(s) = \arg\max_{a \in \mathcal{A}} Q^{*'}(s, a) = \arg\max_{a \in \mathcal{A}} (Q^*(s, a) + \Phi(s)) = \arg\max_{a \in \mathcal{A}} Q^*(s, a) = \pi^*(s)$$

[18] 32.

This theoretical guarantee is a cornerstone of PBRS, distinguishing it from other heuristic shaping methods that might inadvertently alter the optimal policy.

2.10.3 Practical Implications and Design Considerations

The policy invariance property of PBRS offers significant practical advantages:

- **Accelerated Learning:** By providing immediate rewards for progress towards desirable states (e.g., states closer to a goal), PBRS can drastically reduce the sparsity of the reward signal, making exploration more efficient and accelerating convergence, especially in environments with delayed rewards.
- **Reduced Exploration Risk:** Agents are less likely to get stuck in local optima or exhibit undesirable behaviors during early training phases, as the shaping guides them towards more promising regions of the state space.
- **Expert Knowledge Integration:** The potential function $\Phi(s)$ can be designed using expert knowledge about the task. For instance, in a navigation task, $\Phi(s)$ could be inversely proportional to the distance to the goal, providing a positive shaping reward for moving closer to the target. In the V2G context, $\Phi(s)$ could reflect the desirability of states with high battery charge, low grid congestion, or high market prices.

Designing an effective potential function $\Phi(s)$ is key to successful PBRS. Common strategies include:

- **Distance-based Potentials:** For tasks with a clear goal, $\Phi(s)$ can be defined based on the agent's proximity to the goal state (e.g., negative Manhattan distance or Euclidean distance).
- **Subgoal-based Potentials:** In tasks requiring a sequence of steps or subgoals, $\Phi(s)$ can be constructed to provide positive potential for achieving intermediate objectives.
- **Feature-based Potentials:** For complex state spaces, $\Phi(s)$ can be a linear or non-linear function of relevant state features, allowing for more nuanced guidance.

The choice of $\Phi(s)$ should reflect the designer's intuition about what constitutes "progress" or "desirable states" without explicitly dictating the optimal actions.

2.11 Dynamic and Adaptive Rewards

In contrast to PBRS, which typically employs a static potential function throughout training, dynamic or adaptive reward functions are designed to evolve over time. This approach is particularly valuable for complex problems where the relative importance of different objectives may shift as the agent's competency develops, or as the environment itself changes.

2.11.1 Motivation and Mechanisms

Dynamic reward functions offer several advantages:

- **Addressing Evolving Objectives:** In multi-objective problems like V2G, an agent might initially struggle with basic tasks (e.g., maintaining EV charge). As it masters these, the reward function can adapt to emphasize more advanced objectives (e.g., optimizing V2G service provision while avoiding grid overloads).
- **Mitigating Conflicting Goals:** Early in training, conflicting objectives can hinder learning. Dynamic rewards can prioritize certain objectives initially, gradually introducing others as the agent becomes more capable.
- **Responding to Environmental Changes:** In non-stationary environments, an adaptive reward function can adjust its weighting of different components to reflect current conditions (e.g., higher penalty for grid overload during peak demand).

Mechanisms for implementing dynamic rewards include:

- **Time-Varying Weights:** The weights assigned to different components of a composite reward function can be adjusted based on training epochs, agent performance metrics, or predefined schedules.
- **Curiosity-Driven Rewards:** Intrinsic rewards, such as those based on novelty or prediction error, can be dynamically added or removed to encourage exploration in early stages and then faded out as the agent becomes proficient.
- **Adaptive Scaling:** Reward magnitudes can be scaled dynamically to maintain appropriate learning signals as the agent's performance improves or as the range of possible rewards changes.
- **Meta-Learning for Rewards:** More advanced techniques involve meta-learning algorithms that learn to generate or adapt reward functions based on observed agent behavior and task progress.

In the V2G context, an agent might initially receive a high reward for simply connecting to the grid and maintaining a minimum charge level. As training progresses, the reward function could dynamically incorporate larger penalties for grid instability events or higher incentives for profitable energy transactions, thereby guiding the agent towards more sophisticated and holistic V2G management strategies.

2.12 Curriculum Learning

"Can curriculum learning be considered as training the same model on different scenarios, starting with easier ones and gradually moving to more difficult ones?"

Yes, essentially, curriculum learning (CL) can be described as a strategy where the same model is trained on scenarios of increasing difficulty, starting from the easiest and progressing to the most difficult. This general idea, however, is implemented in different ways, as demonstrated by the two reference papers. The research by Pocius et al. focuses on a curriculum based on **task simplification**, whereas Freitag et al. propose a more specific and advanced approach based on **reward function simplification**[19].

2.12.1 Specific Principles and Applications

The core principle of CL is to guide the agent's learning to prevent it from being overwhelmed by the full complexity of the final problem. The provided research offers concrete insights into how and why this approach works. **Pocius et al.** empirically compared curriculum learning, reward shaping, and visual hints in a navigation task within a Minecraft environment[20]. Their curriculum involved training the agent first on a simpler task (navigating a single room) before moving to a more complex one (navigating through two rooms). Their key finding was that, for their specific task, **curriculum learning had the most significant impact on performance**, surpassing the effectiveness of reward shaping. This suggests that for certain problems, structuring the learning experience through progressively harder tasks is a more powerful strategy than finely tuning the immediate reward signal[21]. On the other hand, **Freitag et al.** addressed the problem of complex reward functions with multiple and potentially conflicting terms, which often lead agents to get stuck in local optima (e.g., an agent learning to satisfy a constraint without completing the main objective)[22]. To solve this, they proposed a **two-stage reward curriculum**:

1. **Stage 1:** The agent is trained using only a subset of the reward function, termed the "base reward" (r_b), which encodes the primary task objective.
2. **Stage 2:** Once the agent has sufficiently learned the basic task, the curriculum switches to training on the full reward function, which also includes the constraint terms (r_c).

One of their key innovations is a mechanism to **automatically switch from one stage to the next** by monitoring how well the actor's policy fits the critic's Q-function. They demonstrated that this approach is particularly effective when constraints have a high weight, as it prevents the agent from being "distracted" by the constraints before understanding the main goal, leading to more stable and higher-performing final policies[23].

[19] 15, 36.
[20] 36.
[21] 36.
[22] 15.
[23] 15.

2.12.2 Curriculum Learning in V2G

For the V2G challenge, the two-stage reward curriculum approach proposed by Freitag et al. is particularly suitable, given the multi-objective nature of the problem. A curriculum could be structured as follows:

1. **Stage 1: Basic Charge Management.** The agent is trained with a simplified reward function (the "base reward," r_b) that solely rewards maintaining the charge levels of electric vehicles (EVs) above a minimum threshold. All other objectives, such as grid interaction or profit, are ignored.

2. **Stage 2: Full Optimization.** Once the agent has effectively learned to manage charging, it transitions to the full reward function. This includes the base reward plus the "constraint reward" terms (r_c), which introduce incentives for grid stability, cost minimization, and adherence to user preferences.

This structured approach, as demonstrated by Freitag et al., prevents the agent from being overwhelmed by conflicting objectives from the start, promoting the development of more robust and generalizable V2G management policies[24].

2.13 The Rise of Deep Reinforcement Learning for V2G Control

The convergence of Reinforcement Learning (RL) with the substantial representational capabilities of deep neural networks has given rise to Deep Reinforcement Learning (DRL), which currently represents the leading edge of V2G control research. The development of DRL algorithms has yielded a comprehensive toolkit, primarily divided into two main families: off-policy and on-policy methods, each exhibiting distinct operational characteristics. The following figure illustrates the evolutionary relationships between the algorithms that will be discussed, highlighting how newer ideas were built to overcome the limitations of their predecessors.

2.13.1 Neural Networks as Function Approximators

Neural networks are computational models, inspired by the interconnected structure of neurons in the human brain, designed to recognize complex patterns in data. They are comprised of layers of interconnected nodes, or *neurons*. Each neuron receives inputs, performs a weighted sum of these inputs, adds a bias, and then passes the result through a non-linear activation function to produce an output. This process can be described mathematically for a single neuron as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.8}$$

[24] 15.

Here, y represents the neuron's output, x_i are the inputs, which are multiplied by their corresponding weights w_i. A bias, b, is added to the weighted sum. This entire result is then transformed by an activation function, f. The role of the activation function is crucial as it introduces non-linearity, enabling the network to learn and model complex, non-linear relationships in the data. Common examples of activation functions include the Sigmoid, hyperbolic tangent (tanh), and the Rectified Linear Unit (ReLU).

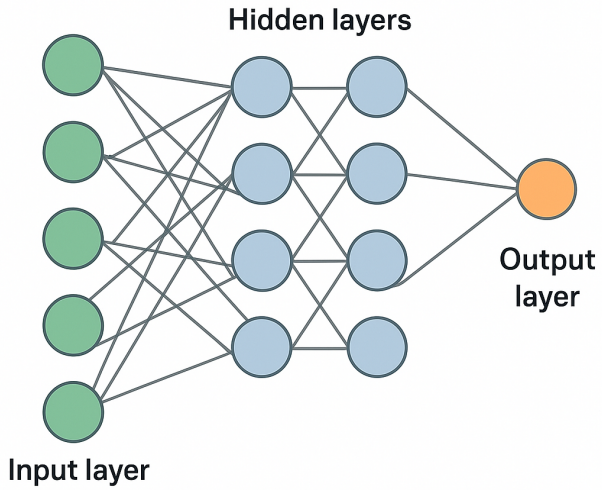


Figure 2.6: Structure of a Neural network

A neural network is constructed with an input layer 2.6, which receives the raw data, one or more *hidden layers* where the computation occurs, and an output layer that produces the final prediction or decision. **When a network contains multiple hidden layers, it is termed a *deep neural network*** (DNN). This depth allows the network to learn a hierarchical representation of features from the data, where each layer learns to identify progressively more complex patterns.

2.14 The Fundamental Role of DNNs in DRL

A prime example of this is the Deep Q-Network (DQN) algorithm, where a DNN is used to approximate the action-value function, $Q(s, a)$. The network receives the environment's state, s, as input and outputs the estimated Q-value for each possible action, a. The network is trained by minimizing a loss function, typically the mean squared error between the predicted Q-value and a target Q-value derived from the Bellman equation. The loss function is given by:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{2.9}$$

In this equation, θ are the weights of the neural network being trained, while θ^- are the weights of a separate, periodically updated target network used to stabilize training. The term $r + \gamma \max_{a'} Q(s', a'; \theta^-)$ is the target value, and the expectation $\mathbb{E}$ is taken over a batch of experiences (s, a, r, s') sampled from a replay memory D. This approach enabled the agent to learn directly from high-dimensional sensory inputs, like raw pixels in Atari games, achieving human-level performance.[25]

Beyond value approximation, deep neural networks are also pivotal in policy gradient methods, where they directly parameterize the agent's policy, π. In this setup, the network, often called a *policy network*, takes a state as input and outputs a probability distribution over the possible actions. The network's weights, θ, are updated by performing gradient ascent on an objective function, $J(\theta)$, which represents the expected cumulative reward. The policy gradient theorem provides a way to update the policy parameters:

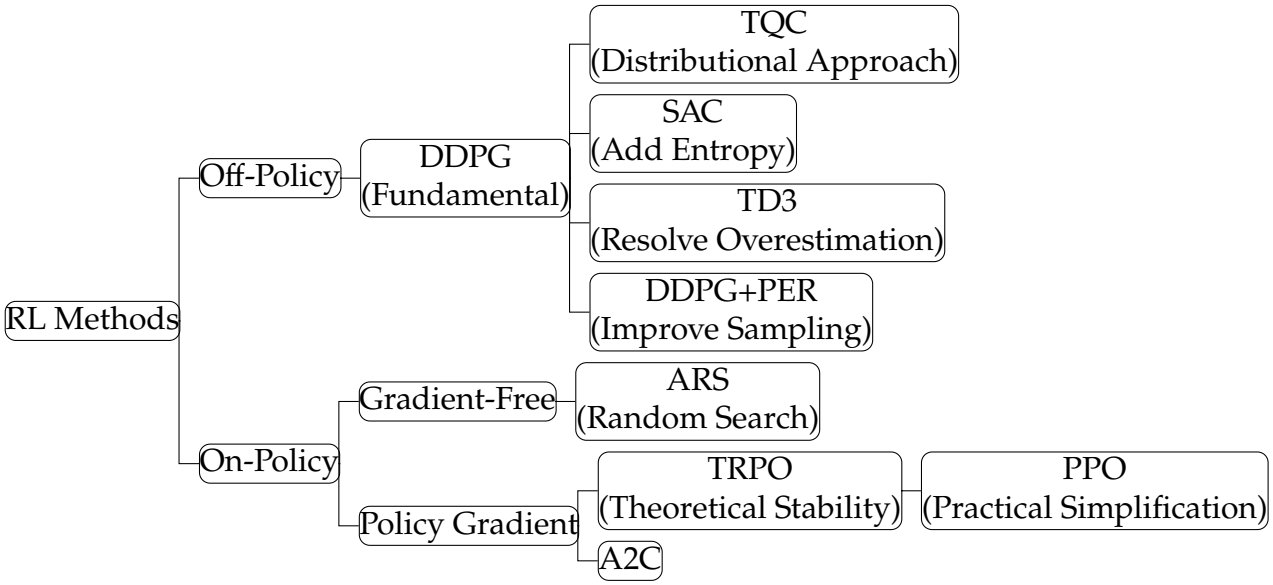
$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right] \tag{2.10}$$

Here, the gradient of the objective function is calculated as the expectation of the sum of the gradients of the log probabilities of the actions taken, each weighted by the cumulative future reward, G_t. This effectively increases the probability of actions that lead to higher returns.[26]

The integration of deep neural networks has thus marked a significant leap in the quality and capability of reinforcement learning. By enabling agents to learn from high-dimensional raw data and generalize across vast state spaces, DRL has unlocked solutions to complex control problems previously considered out of reach, and it now stands as a critical tool for advancing research in V2G control systems.[27]

The convergence of RL with the substantial representational capabilities of deep neural networks has given rise to **Deep Reinforcement Learning (DRL)**, which currently represents the leading edge of V2G control research. The development of DRL algorithms has yielded a comprehensive toolkit, primarily divided into two main families: off-policy and on-policy methods, each exhibiting distinct operational characteristics. The following figure illustrates the evolutionary relationships between the algorithms that will be discussed, highlighting how newer ideas were built to overcome the limitations of their predecessors.

[25] 48.
[26] 45.
[27] 37.



2.14.1 Off-Policy Methods: Data-Efficient Learning from Experience

Off-policy algorithms distinguish themselves through their capacity to learn optimal policies from data generated by different, often more exploratory, behavioral policies. **This enables them to reuse historical experiences stored in large *replay buffers***, breaking temporal correlation between experiences and achieving high sample efficiency.
A pivotal question in this domain is how to balance the exploration of new actions with the exploitation of known good actions. Off-policy methods, by their nature, are well-suited to this challenge.

> Can an agent learn about the optimal way to behave while behaving sub-optimally?

The development of algorithms like Soft Actor-Critic (SAC)[28] provides a compelling, falsifiable hypothesis: that by explicitly encouraging policy entropy,a measure of randomness,an agent can be guided to explore more broadly and avoid collapsing into a narrow, locally optimal policy. This work builds on that premise, testing whether such an approach can robustly discover profitable and stable V2G control strategies in the face of profound uncertainty.

Deep Deterministic Policy Gradient (DDPG)

A foundational algorithm that successfully extended Deep Q-Networks (DQN) to high-dimensional, continuous action spaces, DDPG represented a significant breakthrough for control problems including V2G[29]. It employs an actor-critic architecture where the actor deterministically maps states to actions. However,

[28] 19.
[29] 26.

practical applications often encounter substantial training instability and systematic vulnerability to **overestimation bias**, where critic networks systematically overestimate Q-values, resulting in suboptimal policy learning[30].
Despite these limitations, the foundational concepts of DDPG have been successfully applied and extended in various V2G contexts. For instance, Wang et al.[31] propose a DDPG-based system for demand response management, demonstrating its potential to reduce charging costs.
But can such an approach scale to large, complex systems?
Zhang et al.[32] tackle this question by introducing a transfer learning framework built on DDPG, aiming to coordinate large-scale V2G operations with renewable energy sources. Their work puts forth a testable hypothesis: that knowledge learned in a small-scale environment can be effectively transferred to a large-scale one, thus mitigating the sample inefficiency of DRL. This thesis will implicitly test this hypothesis by evaluating the performance of DRL agents in scenarios of varying complexity.

Twin Delayed DDPG (TD3)

Developed as a direct successor to address DDPG's instabilities, TD3 introduces three key innovations that have become standard in contemporary DRL[33]. It incorporates (1) **clipped double Q-learning**, utilizing paired critic networks and selecting minimum estimates to mitigate overestimation bias; (2) **delayed policy updates**, updating actors less frequently than critics for enhanced stability; and (3) **target policy smoothing**, adding noise to target actions for learning regularization. These enhancements establish TD3 as a more robust and reliable baseline for complex V2G applications[34].
The effectiveness of TD3 in V2G scheduling has been explored by multiple researchers. For example, Dou et al.[35] apply a TD3-based approach to the optimal scheduling of EV charging and discharging, demonstrating its ability to find profitable strategies. But can this profitability be achieved without negatively impacting the grid? Ding et al.[36] investigate this by using TD3 for charging scheduling while considering distribution network voltage stability. Their work provides a clear, falsifiable test of the hypothesis that a DRL agent can learn to balance its own economic incentives with the physical constraints of the power grid. This thesis builds upon this line of inquiry by incorporating explicit penalties for transformer overloads into the reward function, directly testing the agent's ability to learn this trade-off.

[30] 34, 2.
[31] 50.
[32] 54.
[33] 16.
[34] 27, 49.
[35] 3.
[36] 27.

SAC

SAC represents a state-of-the-art off-policy algorithm for continuous control, **recognized for superior sample efficiency and stability**[37]. Its fundamental innovation involves the **maximum entropy framework**. The agent's objective is modified to maximize both expected reward and policy entropy. This entropy bonus encourages agents to act as randomly as possible while maintaining task success, promoting broader exploration, improved noise robustness, and reduced risk of poor local optima convergence.
The principle of maximum entropy, while powerful, raises a critical question:
Does encouraging random behavior compromise the safety and reliability of the system?
Gu et al.[38] directly confront this by proposing a safe reinforcement learning approach that considers battery health. Their work tests the hypothesis that one can achieve the exploration benefits of SAC while simultaneously satisfying hard constraints related to battery degradation. This is a crucial step towards making DRL a viable technology for real-world V2G deployment, and it underscores the importance of the multi-objective reward functions explored in this thesis.

Truncated Quantile Critics (TQC)

TQC addresses overestimation bias through a distributional RL approach[39]. Rather than learning single expected returns (Q-values), its critic **learns complete return probability distributions using quantile regression**.
Digression on Quantile regression:
Quantile regression is an extension of the classical linear regression model, which estimates the conditional mean of a response variable. Instead of focusing only on the mean, quantile regression estimates the conditional quantiles of the response distribution, providing a more complete statistical view. Formally, given data $\{(x_i, y_i)\}_{i=1}^n$ with predictors $x_i \in \mathbb{R}^p$ and response $y_i \in \mathbb{R}$, the τ-th conditional quantile ($0 < \tau < 1$) of Y given $X = x$ is modeled as

$$Q_Y(\tau \mid X = x) = x^\top \beta_\tau,$$

where β_τ are the regression coefficients associated with the quantile τ. The quantile regression estimator is obtained by solving the optimization problem:

$$\hat{\beta}_\tau = \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \rho_\tau\big(y_i - x_i^\top \beta\big),$$

where $\rho_\tau(u)$ is the so-called "check function," defined as

$$\rho_\tau(u) = u \cdot \big(\tau - \mathbf{1}_{\{u<0\}}\big).$$

[37] 18.
[38] 17.
[39] 24.

This asymmetric loss function penalizes underestimation and overestimation differently, depending on the chosen quantile τ. For example, $\tau = 0.5$ corresponds to the conditional median regression.
After learned this can be affirmed that: learning multiple return distribution quantiles and truncating the most optimistic quantile estimates before averaging, it provides a more **principled and effective method for removing primary overestimation bias sources**, frequently achieving superior performance.

Enhancement: Prioritized Experience Replay (PER)

This represents not a standalone algorithm but a crucial orthogonal modification for off-policy methods. Standard replay buffers sample past transitions uniformly. PER samples transitions based on their "importance," typically proportional to TD error magnitude[40]. This focuses learning processes on surprising or informative experiences, significantly accelerating convergence and improving final performance.

2.14.2 On-Policy Methods: Stability through Cautious Updates

On-policy methods learn exclusively from data generated by current policies being optimized. Once data is used for updates, it is discarded. While this makes them inherently less sample-efficient than off-policy counterparts, their updates often demonstrate greater stability and reduced divergence risk.

Advantage Actor-Critic (A2C/A3C)

A2C represents a foundational synchronous, on-policy Actor-Critic algorithm. Its powerful extension, **Asynchronous Advantage Actor-Critic (A3C)**, was a landmark contribution demonstrating parallelism benefits[41]. A3C utilizes multiple parallel workers, each with individual model and environment copies. These workers interact with their environments independently, and collected gradients update a global model asynchronously. This decorrelates data streams and provides powerful stabilizing effects on learning processes.
While often considered less sample-efficient than their off-policy counterparts, on-policy methods like A2C and its variants are still actively explored in the V2G domain. For example, Chifu et al.[42] propose a Deep Q-Learning based approach for smart scheduling of EVs for demand response, which shares some of the on-policy characteristics. Their work raises the question:
Can the stability and reliability of on-policy training outweigh the sample efficiency of off-policy methods in a problem where the environment is highly stochastic? This thesis provides a direct comparison by benchmarking several state-of-the-art off-policy agents against on-policy baselines, allowing for an empirical test of this long-standing hypothesis in the DRL community.

[40] 41.
[41] 31.
[42] 11.

Trust Region Policy Optimization (TRPO)

TRPO was the first algorithm to formalize policy update size constraints for guaranteed monotonic policy improvement[43]. It maximizes a "surrogate" objective function subject to policy change constraints, measured by Kullback-Leibler (KL) divergence. This creates a "trust region" within which new policies are guaranteed to improve upon previous ones, preventing catastrophic updates that can permanently derail learning. However, implementation complexity arises from second-order optimization requirements.

Proximal Policy Optimization (PPO)

PPO achieves TRPO's stability benefits and reliable performance using only first-order optimization, making it far simpler to implement and more broadly applicable[44]. It employs a novel **clipping** mechanism in its objective function to discourage large policy updates that would move new policies too far from previous ones, effectively creating "soft" trust regions. Due to its excellent balance of performance, stability, and implementation simplicity, PPO has become a default choice for many on-policy applications.

2.14.3 Gradient-Free Methods: An Alternative Path

Augmented Random Search (ARS)

As a counterpoint to dominant gradient-based methods, ARS represents a gradient-free approach that optimizes policies directly in parameter space[45]. It operates by exploring random policy parameter perturbations and updating central policy parameter vectors based on observed performance of these perturbations. While often less sample-efficient for complex, high-dimensional problems, its extreme simplicity, scalability, and robustness to noisy rewards can make it competitive in certain domains.

2.15 The Model-Based Benchmark: Model Predictive Control (MPC)

While DRL offers powerful model-free approaches, model based techniques requires benchmarking against its most robust model-based counterpart: **Model Predictive Control (MPC)**. MPC originated in the 1970s within the process control industry, with pioneering contributions by Richalet et al.[46] and Cutler and Ramaker[47]. The theoretical foundations for stability and optimality were subse-

[43] 43.
[44] 42.
[45] 28.
[46] 21.
[47] 13.

quently established rigorously by researchers including Mayne and Rawlings[48]. At its core, MPC represents a proactive, forward-looking strategy that employs explicit mathematical system models to predict future evolution. At each control step, it solves finite-horizon optimal control problems to determine optimal control action sequences. Its primary strength, explaining widespread industrial adoption, lies in its inherent capability to proactively handle complex system dynamics and operational constraints[49].

2.16 Model Predictive Control Formulation

Section based on "Predictive Control for Linear and Hybrid Systems" by F. Borrelli, A. Bemporad, and M. Morari.

2.16.1 The Finite Time Optimal Control Problem

At each time step t, given the current state measurement $x(t)$, a Model Predictive Control (MPC) law is defined by solving a constrained finite time optimal control problem. This is often referred to as Receding Horizon Control (RHC). Consider the discrete-time linear time-invariant system:

$$x(k+1) = Ax(k) + Bu(k) \tag{2.11}$$

The optimization problem solved at the current time t is formulated as follows, using $x(t)$ as the initial state x_0:

$$J_0^*(x(t)) = \min_{U_0} J_0(x(t), U_0) \tag{2.12}$$

where the cost function J_0 for a quadratic objective is defined as:

$$J_0(x(0), U_0) = x_N' P x_N + \sum_{k=0}^{N-1} (x_k' Q x_k + u_k' R u_k) \tag{2.13}$$

The minimization is subject to the following constraints for $k = 0, \ldots, N-1$:

$$\begin{aligned} \text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k, && (2.14)\\ & x_k \in \mathcal{X}, && (2.15)\\ & u_k \in \mathcal{U}, && (2.16)\\ & x_N \in \mathcal{X}_f, && (2.17)\\ & x_0 = x(t). && (2.18) \end{aligned}$$

Here, the variables are defined as:

- $x_k \in \mathbb{R}^n$ denotes the state vector at time k obtained by starting from the state $x_0 = x(t)$ and applying the input sequence $u_0, \ldots, u_{k-1}$.

[48] 29.
[49] 30.

- $U_0 = [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^{mN}$ is the decision vector containing all future inputs over the prediction horizon N.
- P, Q are positive semi-definite state penalty matrices, and R is a positive definite input penalty matrix.
- $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are polyhedra representing state and input constraints, respectively.
- $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a terminal polyhedral region that the state x_N is constrained to enter.

2.16.2 The Receding Horizon Policy

The optimization problem yields an optimal sequence of control inputs $U_0^*(x(t)) = \{u_0^*, u_1^*, \ldots, u_{N-1}^*\}$. In the receding horizon control strategy, only the first element of this sequence is applied to the system:

$$u(t) = u_0^*(x(t)) \tag{2.19}$$

At the next time step, $t+1$, a new state measurement $x(t+1)$ is obtained. The entire optimization problem is then solved again over a shifted horizon $[t+1, t+1+N]$, using $x(t+1)$ as the new initial state. This process is repeated at each sampling instant.

2.16.3 Implicit MPC: Online Optimization

The most common formulation involves **Implicit MPC**, where constrained optimization problems are solved online at each control step. For linear systems with quadratic costs, this typically constitutes a Quadratic Program (QP). The controller's objective involves finding future control input sequences

$$U = [u_{t|t}, \ldots, u_{t+N-1|t}]$$

that minimize a cost function J over a prediction horizon N. A key insight from[50] is that in this formulation, the control law is defined **implicitly** as the result of an optimization problem. There is no pre-computed algebraic function; the control action is discovered at each step through a numerical computation. This approach is powerful due to its directness but is fundamentally limited by the need to perform this computation online. The authors of[51] highlight this on page xii, stating:

> "One limitation of MPC is that running the optimization algorithm on-line at each time step requires substantial time and computational resources."

[50] 8.
[51] 8.

The finite-horizon optimal control problem is formulated as:

$$\min_{U_t} J(x_t, U_t) = \sum_{k=0}^{N-1} \left(x_{t+k|t}^\top \mathbf{Q} x_{t+k|t} + u_{t+k|t}^\top \mathbf{R} u_{t+k|t} \right) + x_{t+N|t}^\top \mathbf{P} x_{t+N|t} \tag{2.20}$$

where $x_{t+k|t}$ represents the predicted state at future step k based on information at time t, and $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{P}$ are weighting matrices defining trade-offs between state deviation and control effort. This optimization is subject to system dynamics and operational constraints:

$$x_{t+k+1|t} = \mathbf{A} x_{t+k|t} + \mathbf{B} u_{t+k|t} \tag{2.21}$$

$$x_{\min} \leq x_{t+k|t} \leq x_{\max} \tag{2.22}$$

$$u_{\min} \leq u_{t+k|t} \leq u_{\max} \tag{2.23}$$

At each time step t, this complete problem is solved, but only the first action of the optimal sequence, $u_{t|t}^*$, is applied to the system. The process then repeats at the next time step, $t+1$, using new system state measurements. This feedback mechanism, known as a *receding horizon* strategy, makes MPC robust to disturbances and model mismatch[52].

2.17 Improvement of the Standard Fixed-Horizon MPC Formulation

Model Predictive Control (MPC) addresses the control of a discrete-time nonlinear system, as described in[53]:

$$x^+ = f(x, u) \tag{2.24}$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the control input. The objective is to solve a finite-horizon optimal control problem at each time step. For a **fixed prediction horizon** N, the problem is formulated as finding a control sequence $u_N = (u(0), \ldots, u(N-1))$ that minimizes a cost function, defined in eq. (11) of the paper as:

$$V_N(x) = \sum_{k=0}^{N-1} l(x(k), u(k)) + V_f(x(N)) \tag{2.25}$$

This minimization is subject to system dynamics, state and input constraints, and the crucial terminal constraint $x(N) \in \mathcal{X}_f$. Here, $\mathcal{X}_f$ is a terminal set where stability is guaranteed, and $V_f(x)$ is a terminal cost that acts as a control Lyapunov function on $\mathcal{X}_f$[54].

[52] 10.
[53] 23.
[54] 23.

2.17.1 Limitation of the Fixed-Horizon Approach

The primary drawback of this standard formulation is the static nature of the horizon N. The choice of N represents a difficult compromise. As Krener points out on page 2, "One would expect when the current state x is far from the operating point, a relatively long horizon N is needed... but as the state approaches the operating point shorter and shorter horizons can be used"[55]. A fixed horizon long enough for the worst-case scenario is computationally wasteful for the majority of the operational time, as it increases the dimensionality of the optimization problem ($m \times N$) solved online.

2.18 Improving the Formulation with Adaptive Horizon MPC (AHMPC)

The core innovation of Adaptive Horizon Model Predictive Control (AHMPC) is to make the horizon length state-dependent, adapting it online to be as short as possible while maintaining stability.

2.18.1 The Ideal (but Impractical) AHMPC Scheme

The paper first introduces an "ideal" version of AHMPC. This scheme relies on a function $N(x)$, defined as the minimum integer N for which a feasible control sequence exists that drives the state x into the terminal set $\mathcal{X}_f$ (see Assumption 4 in[56]). The resulting optimal cost and control law are then state-dependent through this horizon:

$$V(x) = V_{N(x)}(x) \tag{2.26}$$

$$\kappa(x) = \kappa_{N(x)}(x) \tag{2.27}$$

As shown in Section II of the paper, this formulation leads to a valid Lyapunov function, confirming its stabilizing properties. However, this scheme is impractical because "in general, it is impossible to compute the function $N(x)$"[57].

2.18.2 The Practical AHMPC Algorithm

Section III of the paper presents a practical algorithm that circumvents the need to know $N(x)$ or even the terminal set $\mathcal{X}_f$ explicitly. The key idea is to use the terminal controller $\kappa_f(x)$ and terminal cost $V_f(x)$ to *verify online* if the chosen horizon is sufficient.
At a given state x and with a current horizon guess N, the algorithm is as follows:

[55] 23.
[56] 23.
[57] 23.

1. **Solve and Extend:** Solve the N-horizon optimal control problem to get the trajectory $x^0(\cdot)$. Then, extend this trajectory for L additional steps using the terminal feedback law:

$$x^0(k+1) = f(x^0(k), \kappa_f(x^0(k))), \quad \text{for } k = N, \ldots, N+L-1 \tag{2.28}$$

2. **Verify Stability Conditions:** Check if the Lyapunov conditions, given as equations (18) and (19) in[58], hold along this extended part of the trajectory:

$$V_f(x^0(k)) \geq \alpha(|x^0(k)|) \tag{2.29}$$

$$V_f(x^0(k)) - V_f(x^0(k+1)) \geq \alpha(|x^0(k)|) \tag{2.30}$$

for $k = N, \ldots, N+L-1$. Failure to satisfy these conditions implies that the terminal state $x^0(N)$ is likely not in a region of stability.

3. **Adapt the Horizon:** The adaptation logic is described on page 5 of the paper:

   - **If conditions hold:** The horizon N is considered sufficient. The control $u^0(0)$ is applied. At the next state x^+, the controller attempts a shorter horizon, typically $N-1$.
   - **If conditions fail:** The horizon N is too short. The controller "increase[s] N by 1 and... solve[s] the optimal control problem over the new horizon" from the same state x[59]. This is repeated until a sufficient horizon is found.

2.18.3 Advantages of the AHMPC Formulation

This practical scheme offers significant advantages, as highlighted in the paper's conclusion:

- **Computational Efficiency:** The primary advantage is that "the AHMPC horizon length decreases as the process is stabilized thereby lessening the on-line computational burden"[60].
- **No Explicit Terminal Set:** The algorithm cleverly "proceeds without knowing the minimum horizon length function $N(x)$ and without knowing the domain of Lyapunov stability of the terminal cost $V_f(x)$"[61]. This removes a major hurdle in traditional MPC design.

2.19 Further Enhancements via Learning-Based Approaches

While the practical AHMPC algorithm provides a significant advantage, it still relies on an iterative, trial-and-error process of increasing N when the current

[58] 23.
[59] 23.
[60] 23.
[61] 23.

horizon is insufficient. This can introduce computational delays.
Modern data-driven techniques, such as Deep Reinforcement Learning (DRL), offer a path to mitigate this. DRL has proven effective for learning complex control policies for challenging problems, such as managing Electric Vehicle charging in volatile markets[62]. A hybrid approach could be envisioned where a DRL agent is trained offline to approximate the ideal horizon function $N(x)$. This learned function would provide a highly accurate initial guess for the horizon at each step, potentially eliminating the need for online iterative adjustments and combining the computational speed of a learned policy with the rigorous stability and constraint-handling framework of AHMPC.

2.20 Explicit MPC: Offline Pre-computation

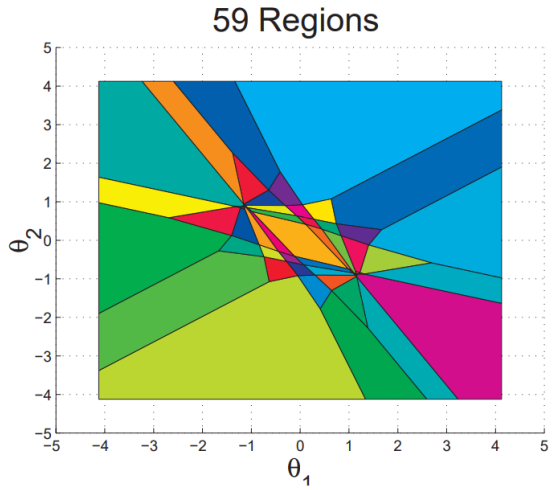


Figure 2.7: Example of a structure of a polyhedral region by Alberto Bemporad and Carlo Filippi

[63] For systems with fast dynamics or limited online computational capacity, **Explicit MPC** offers an alternative approach. The key idea is to move the computational effort entirely offline. As detailed extensively in[64], this is achieved by treating the current state x_t not as a fixed value, but as a vector of parameters. The constrained optimization problem is then solved offline for all possible initial states within a given range using **multi-parametric programming**.
The authors of[65] frame this as the main contribution of their work (Preface, page xii):

[62] 34.
[63] 5.
[64] 8.
[65] 8.

> "we want to determine the [...] feedback control law $f(x)$ that generates the optimal $u_k = f(x(k))$ **explicitly** and not just **implicitly** as the result of an optimization problem."

The result is not an online algorithm, but a pre-computed, explicit control law, $K(x_t)$, which for linear systems is a **piecewise affine (PWA) function** of the state vector x_t:

$$u^*(x_t) = \mathbf{F}_i x_t + \mathbf{g}_i \quad \text{if } x_t \in \mathcal{X}_i \tag{2.31}$$

The state space is partitioned into a set of convex polyhedral regions $\mathcal{X}_i$, with a unique affine control law defined for each region. Online operation is thereby reduced from solving a QP to a simple function evaluation: first, a fast lookup operation identifies which region $\mathcal{X}_i$ the current state x_t occupies, and second, the corresponding simple affine control law is applied. This trades a very high offline computational burden and significant memory requirements for extremely fast and deterministic online execution[66].

2.20.1 Deep Learning for an Efficient Explicit MPC Representation

While implicit MPC provides an optimal control action, its reliance on solving a numerical optimization problem at each sampling time makes it unsuitable for systems with fast dynamics. An alternative is Explicit MPC, where the control law—a Piecewise Affine (PWA) function of the state—is pre-computed offline. However, this approach suffers from a significant drawback: the number of polyhedral regions, and thus the memory required to store the corresponding affine laws, can grow exponentially with the prediction horizon and the number of constraints[67]. This "curse of dimensionality" severely limits its practical application, especially in memory-constrained embedded systems.
To address this challenge, the work in[68] proposes a novel approach that leverages deep learning to create a highly efficient representation of the explicit MPC law. The methodology bridges the gap between the implicit and explicit approaches by using the former to train the latter.

Training via Implicit MPC Data Generation

The core idea is to train a deep neural network to act as the explicit controller. This requires a comprehensive dataset of optimal state-action pairs, which is generated in a crucial offline phase. As detailed in[69], this is achieved by repeatedly using the **implicit MPC solver**:

1. A large number of state points ($x_{tr,i}$) are sampled from the relevant operating region of the system.

[66] 4, 8.
[67] 22.
[68] 22.
[69] 22.

2. For each state point, the full MPC optimization problem (a Quadratic Program, or QP) is solved to find the corresponding optimal control input ($u^*_{tr,i}$). This is precisely the task an implicit MPC controller performs online.

3. The resulting pairs $(x_{tr,i}, u^*_{tr,i})$ form a large dataset that effectively maps states to their optimal control actions.

This dataset is then used to train a deep neural network via supervised learning, minimizing the error between the network's output and the optimal control actions. In essence, the computationally intensive implicit solver is used offline to "teach" a fast and compact neural network how to behave optimally.

Exact PWA Representation with Deep ReLU Networks

The power of this method stems from the theoretical insight that a deep neural network with Rectified Linear Unit (ReLU) activation functions is not merely an approximator but can, in fact, **exactly represent** the PWA function that defines the explicit MPC feedback law[70].
The foundation of this exact representation lies in two main results. First, any scalar PWA function $K_i(x_t)$ (representing the i-th component of the control vector) can be expressed as the difference of two convex PWA functions:

$$K_i(x_t) = \gamma_i(x_t) - \eta_i(x_t) \tag{2.32}$$

Second, any convex PWA function, which can be described as the pointwise maximum of a set of affine functions, can be exactly represented by a deep ReLU network. Specifically, a convex function composed of N affine regions can be perfectly modeled by a network with width $n_x + 1$ (where n_x is the state dimension) and depth N[71]. Combining these findings, the complete multi-output MPC control law $K(x_t)$ can be exactly represented by a vector of n_u pairs of deep neural networks, where each pair models the γ_i and η_i components for a single control output:

$$K(x_t) = \begin{bmatrix} \mathcal{N}(x_t; \theta_{\gamma,1}) - \mathcal{N}(x_t; \theta_{\eta,1}) \\ \vdots \\ \mathcal{N}(x_t; \theta_{\gamma,n_u}) - \mathcal{N}(x_t; \theta_{\eta,n_u}) \end{bmatrix} \tag{2.33}$$

where $\mathcal{N}$ denotes a deep ReLU network with its corresponding set of parameters θ. The particular advantage of using *deep* neural networks lies in their representational efficiency. While the number of parameters (and thus, the memory footprint) of a deep network grows linearly with its depth, the number of linear regions it can represent can grow exponentially[72]. This provides an inverse relationship to the problem of traditional Explicit MPC, allowing a deep network to capture an exponentially large number of control regions with only a modest, linearly growing memory cost. Consequently, this deep learning-based representation transforms the explicit MPC law into a highly compact and fast-to-evaluate

[70] 22.
[71] 22.
[72] 22.

form, overcoming the primary obstacle of prohibitive memory requirements and enabling the deployment of complex predictive controllers on embedded systems.

2.21 A Comparative Perspective on Control Methodologies

While DRL represents the cutting edge, contextualizing it within the broader control strategy landscape remains crucial for academic rigor. The choice between learning-based, model-free approaches and optimization-based, model-based approaches represents fundamental philosophical and practical trade-offs.

Table 2.1: Comparative Analysis: DRL vs. Model Predictive Control (MPC) for V2G

Aspect	Deep Reinforcement Learning (DRL)	Model Predictive Control (MPC)
Paradigm	Model-Free, learning-based. Learns an optimal policy (a reactive function) via trial-and-error interaction with the environment.	Model-Based, optimisation-based. Solves a constrained optimisation problem at each time step based on a system model and forecasts.
Strengths	• Highly robust to uncertainty and unmodelled stochasticity. • Does not require an explicit, accurate system model. • Can learn complex, non-linear control policies. • Extremely fast inference time (a single forward pass) once trained.	• Explicitly and rigorously handles hard constraints, providing safety guarantees. • Proactive and anticipatory if forecasts are accurate. • Well-established, mature, and theoretically understood.
Weaknesses	• Can be highly sample-inefficient during the training phase. • Lacks hard safety guarantees (an active and important area of research). • The "black box" nature of neural network policies can make them difficult to interpret or verify.	• Performance is fundamentally shackled to the accuracy of the system model and external forecasts. • Computationally expensive at each time step, suffering from the "curse of dimensionality" with system size. • Can be brittle to unexpected forecast errors and unmodelled dynamics.
V2G Suitability	Excellent for dynamic, highly uncertain environments with complex, non-linear trade-offs and a large number of assets.	Good for problems with simple, well-defined dynamics and reliable forecasts, but struggles with the real-world stochasticity and scale of V2G.

Model Predictive Control (MPC) stands as the most powerful model-based alternative. Its primary and most compelling strength lies in its native ability to handle hard constraints, which is critical for ensuring safe operation (such as never violating transformer limits or user energy requirements). However, its performance remains fundamentally constrained by the accuracy of its internal model

and forecasts of future disturbances (prices, user behavior)[73]. In practice, creating accurate, tractable models for the entire V2G domain proves nearly impossible due to non-linear battery dynamics, extreme market volatility, and the profound unpredictability of human behavior. Furthermore, as EV fleet sizes grow, optimization problem dimensionality explodes, making online computation required at each time step intractable for large fleets.
Other methods, such as **meta-heuristic algorithms** (including genetic algorithms and particle swarm optimization), are sometimes proposed. However, these are typically employed for offline scheduling and lack the real-time, reactive responsiveness required for dynamic V2G control in rapidly changing environments.
Ultimately, DRL's singular advantage lies in its ability to learn and internalize the complex, non-linear trade-offs of multi-objective V2G problems directly from data, without requiring explicit models. This makes it uniquely suited to navigating the profound uncertainties of real-world operations. While other methods have their applications, DRL stands out as the most promising technology for deploying truly intelligent, autonomous, and scalable V2G management systems required to achieve the ambitious energy and climate objectives of the European Union.

2.22 A Primer on Lithium-Ion Battery Chemistries and Degradation

The effectiveness, safety, and economic viability of any V2G strategy remain fundamentally constrained by the physical and chemical characteristics of vehicle batteries. Battery chemistry selection dictates EV operational envelopes, influencing energy density, power capabilities, lifespan, and safety profiles. Clear understanding of these trade-offs proves essential for developing robust, realistic, and responsible control algorithms.

2.22.1 Fundamental Concepts and Degradation Mechanisms

Battery degradation is a complex and irreversible process that gradually reduces a cell's ability to store and deliver energy. It manifests primarily as capacity loss (energy fade) and as an increase in internal resistance (power fade). These phenomena are usually attributed to two categories of mechanisms: *calendar aging* and *cyclic aging*[74].
Calendar aging occurs while the battery is at rest, regardless of whether it is fully charged or nearly empty. The main mechanism behind this process is the slow growth of the **Solid Electrolyte Interphase (SEI)** layer on the graphite anode. While a thin and stable SEI layer is necessary for battery operation, its continuous thickening consumes both active lithium and electrolyte, resulting in irreversible capacity loss. The rate of SEI growth is particularly sensitive to operating conditions. High temperatures accelerate chemical reaction rates, including parasitic

[73] 14.
[74] 6.

ones, making storage in hot environments detrimental. Similarly, high states of charge (SoC) correspond to low anode potentials, which increase the reactivity of graphite with the electrolyte and foster faster SEI growth[75]. For this reason, leaving electric vehicles stored for long periods at 100% SoC is generally discouraged. **Cyclic aging**, in contrast, results from the processes that take place during charging and discharging. This form of degradation is of particular importance for V2G applications, which inherently involve frequent cycling. Several mechanisms contribute to cyclic aging. The repeated intercalation and de-intercalation of lithium ions induce mechanical stress due to the expansion and contraction of electrode materials. Over time, this stress may generate micro-cracks in electrode particles, leading to a loss of electrical contact and a reduction of active material, effects that become more severe with larger **Depths of Discharge (DoD)**. Mechanical volume changes can also compromise the integrity of the SEI layer, causing cracks that expose fresh anode surface to the electrolyte and trigger renewed SEI growth. In addition, under demanding conditions,such as very high charging rates (high C-rates) or low temperatures,lithium ions may deposit as metallic lithium on the anode surface rather than intercalating properly. This phenomenon, known as lithium plating, is particularly harmful: it leads to rapid capacity loss and may produce dendritic structures capable of piercing the separator, thereby creating internal short circuits and serious safety hazards[76].
Both calendar and cyclic aging are unavoidable, the latter is especially critical for V2G systems. Since these applications require frequent and sometimes deep cycling, understanding and mitigating cyclic degradation is paramount to guarantee long-term system viability.

2.22.2 Key Automotive Chemistries

The EV market is dominated by several key lithium-ion battery families, primarily distinguished by cathode materials. Each chemistry presents different balances of performance characteristics.

- **Lithium Nickel Manganese Cobalt Oxide (NMC):** For years, this has been the most popular choice due to its excellent balance of energy density, power, and cycle life. The ratio of Nickel, Manganese, and Cobalt can be adjusted (such as NMC111, NMC532, NMC811) to prioritize either energy density (high Nickel) or safety and longevity (lower Nickel).
- **Lithium Nickel Cobalt Aluminum Oxide (NCA):** Offers very high specific energy (energy density by weight), enabling longer vehicle range. However, it typically comes at the cost of slightly lower cycle life and narrower safety margins compared to NMC, requiring more sophisticated thermal management.
- **Lithium Iron Phosphate (LFP):** Is rapidly gaining market share, particularly in standard-range vehicles, due to lower cost (no cobalt, an expensive and

[75] 47.
[76] 6.

ethically contentious material) and superior safety. LFP batteries offer exceptional cycle life (often 3-4 times that of NMC/NCA) and are considered the safest common Li-ion chemistry due to high thermal stability. Their main drawback involves lower energy density.

- **Lithium Titanate Oxide (LTO):** This represents a more niche chemistry using titanate-based anodes instead of graphite. This provides exceptional safety (virtually no thermal runaway risk), extremely long cycle life (>10,000 cycles), and excellent low-temperature performance. However, very low energy density and high cost currently limit its use to specialized applications.

2.22.3 Voltage Profiles and the Challenge of SoC Estimation

The relationship between battery open-circuit voltage and SoC represents a critical, non-linear function unique to each chemistry. The derivative of cell voltage with respect to DoD, $\frac{dV_{cell}}{d(DoD)}$, constitutes a key parameter for Battery Management Systems (BMS). Steep, monotonic slopes allow BMS to accurately infer SoC from simple voltage measurements. Conversely, flat slopes make this estimation extremely difficult.

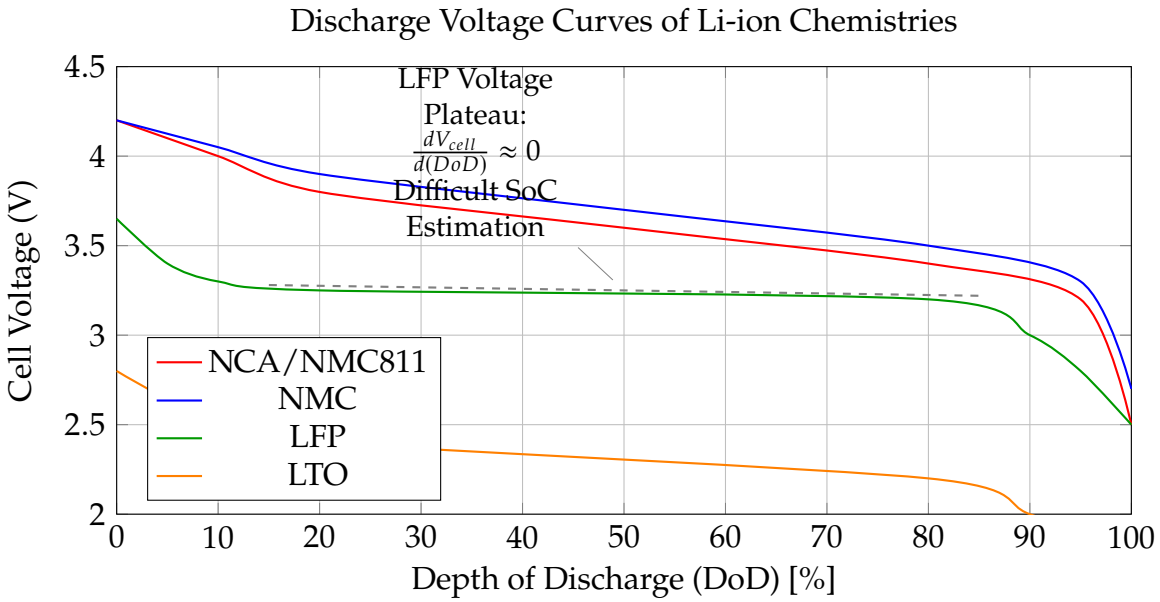


Figure 2.8: Typical discharge voltage curves for various lithium-ion chemistries. The extremely flat profile of LFP makes accurate SoC estimation challenging based on voltage alone, necessitating more complex estimation techniques like Coulomb counting and periodic recalibration.

As shown in Figure 2.8, LFP's remarkably flat voltage plateau makes it nearly impossible for BMS to determine precise SoC in the central operating range (from approximately 20% to 80%) using voltage alone. This necessitates more complex

estimation techniques, such as Coulomb counting (integrating current over time), which can suffer from drift. To correct this drift, LFP-equipped vehicles require periodic full charges to 100% for BMS recalibration. This represents an important operational constraint that V2G control strategies must consider.

2.22.4 Comparative Analysis and Safety Considerations

The trade-offs between chemistries are summarized in Table 2.2. Safety remains paramount, with the primary risk being thermal runaway,a dangerous, self-sustaining exothermic reaction. This risk relates directly to cathode material chemical and thermal stability. Higher energy density generally means more energy packed into smaller mass, which can be released violently if cells are compromised. Consequently, critical temperatures for initiating thermal runaway are generally lower for higher energy density chemistries. LFP's stable phosphate-based structure makes it far more resistant to thermal runaway than nickel-based counterparts, a key reason for its growing popularity.

Table 2.2: Comparative analysis of key automotive battery chemistries, highlighting the trade-offs between performance and safety.

Metric	NCA	NMC	LFP	LTO	LCO
Energy Density (Wh/kg)	200 - 260 (Highest)	150 - 220 (High)	90 - 160 (Moderate)	60 - 110 (Low)	150-200 (High)
Cycle Life	1000 - 2000	1000 - 2500	2000 - 5000+	>10,000	500 - 1000
Safety	Good	Very Good	Excellent	Excellent	Poor
Thermal Runaway Temp (°C)	~150 - 180	~180 - 210	~220 - 270	> 250	~150

Chapter 3

An Enhanced V2G Simulation Framework for Robust Control

Developing, validating, and benchmarking advanced control algorithms for Vehicle-to-Grid (V2G) systems is a complex endeavor. Real-world experimentation is often impractical due to prohibitive costs, logistical challenges, and risks to grid stability and vehicle hardware. To bridge the gap between theory and practice, a realistic, flexible, and standardized simulation environment is a scientific necessity. This thesis builds upon the foundation of **EV2Gym**, a state-of-the-art, open-source simulator designed for V2G smart charging research[1]. This work, however, extends the original framework significantly, transforming it into a high-fidelity **digital twin** engineered not just for single-scenario optimization, but for the development and rigorous evaluation of **robust, generalist control agents**. This enhanced framework offers a two-pronged approach to experimentation: it allows for deep-dive analysis of agents specialized for a single environment, while also introducing a novel methodology for training and testing agents designed to generalize across a multitude of diverse, unpredictable scenarios. This chapter provides an in-depth tour of this extended architecture, its data-driven models, and its unique evaluation capabilities, establishing the methodological bedrock for the rest of this work.

3.1 Core Simulator Architecture

The framework is built on the modular architecture of EV2Gym, which mirrors the key entities of a real-world V2G system. Its foundation on the OpenAI Gym (now Gymnasium) API is a cornerstone, providing a standardized agent-environment interface defined by the familiar language of states, actions, and rewards[2].

[1] 34.
[2] 9.

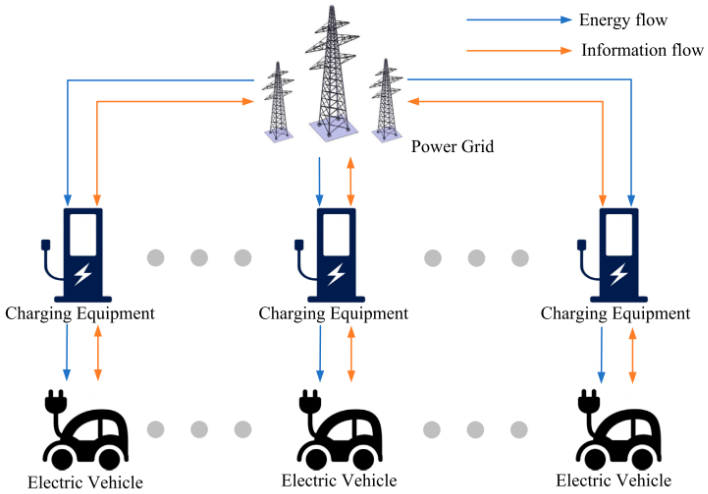


Figure 3.1: Diagram of charging and discharging scheduling for EVs.[3].

The architecture consists of several interacting components:

- **Charge Point Operator (CPO):** The central intelligence of the simulation, managing the charging infrastructure and serving as the primary interface for the control algorithm (the DRL agent). The CPO aggregates system state information and dispatches control actions to individual chargers.
- **Chargers:** Digital representations of physical charging stations, configurable by type (AC/DC), maximum power, and efficiency. This allows for the simulation of heterogeneous charging infrastructures.
- **Power Transformers:** These components model the physical connection points to the grid, aggregating the electrical load from multiple chargers. Crucially, they enforce the physical power limits of the local distribution network and can model inflexible base loads (e.g., buildings) and local renewable generation (e.g., solar panels).
- **Electric Vehicles (EVs):** Dynamic and autonomous agents, each defined by its unique battery capacity, power limits, current and desired energy levels, and specific arrival and departure times.

The simulation process follows a reproducible three-phase structure: (1) **Initialization** from a comprehensive YAML configuration file, (2) a discrete-time **Simulation Loop** where the agent interacts with the environment, and (3) a final **Evaluation and Visualization** phase that generates standardized performance metrics.

3.1.1 Software Implementation and Project Structure

While the conceptual architecture describes the simulator's components, the practical implementation is organized within a modular Python package named

`ev2gym`. This structure promotes code reusability and a clear separation of concerns. The high-level experimentation scripts, such as `run_experiments.py` and `train_mpc_approximator.py`, reside in the project's root directory and act as orchestrators, utilizing the core functionalities provided by the `ev2gym` package.

The key subdirectories within the `ev2gym` package are:

- `baselines/`: This directory contains the implementations for all non-RL controllers. This includes rule-based heuristics (in `heuristics.py`) and, crucially, all variants of the Model Predictive Controllers (in `pulp_mpc.py`).
- `rl_agent/`: This is the central hub for all Reinforcement Learning logic. It contains modules for state vector construction (`state.py`), the library of available reward functions (`reward.py`), and the implementation of custom RL algorithms (`custom_algorithms.py`).
- `utilities/`: A collection of helper functions and utility classes that are used across the entire framework.
- `models/`: This directory is designated for storing the serialized, pre-trained machine learning models, such as the Random Forest model used by the Approximate-Explicit MPC.

This modular software design allows for the independent development and testing of different components, such as control algorithms and reward functions, while maintaining a consistent and unified simulation environment.

3.2 Core Physical Models

The simulation's fidelity is anchored in its detailed, empirically validated models, which are essential for developing control strategies robust enough for real-world application.

3.2.1 EV Model and Charging/Discharging Dynamics

The framework implements a realistic two-stage charging/discharging model that captures the non-linear behavior of lithium-ion batteries, simulating both the **constant current (CC)** and **constant voltage (CV)** phases. Each EV is defined by a rich parameter set: maximum capacity (E_{max}), a minimum safety capacity (E_{min}), separate power limits for charging and discharging ($P_{ch}^{max}, P_{dis}^{max}$), and distinct efficiencies for each process (η_{ch}, η_{dis}).

3.2.2 Battery Degradation Model

To address the critical issue of battery health in V2G operations, the simulator incorporates a semi-empirical battery degradation model. It quantifies capacity

loss (Q_{lost}) as the sum of two primary aging mechanisms[4]: calendar aging and cyclic aging.

- **Calendar Aging (d_{cal}):** Time-dependent capacity loss, influenced by the battery's average State of Charge (SoC) and temperature (Θ). The formula is given by:

$$d_{cal} = 0.75 \cdot (\epsilon_0 \cdot \overline{SoC} - \epsilon_1) \cdot e^{-\epsilon_2/\Theta} \cdot \frac{t_{days}}{(t_{days}+1)^{0.25}} \tag{3.1}$$

- **Cyclic Aging (d_{cyc}):** Wear resulting from charge/discharge cycles, dependent on energy throughput ($E_{exchanged}$), depth-of-cycle (implicitly via $\overline{SoC}$), and the total accumulated charge (Q_{acc}). The formula is:

$$d_{cyc} = (\zeta_0 + \zeta_1 \cdot |\overline{SoC} - 0.5|) \cdot \frac{E_{exchanged}}{\sqrt{Q_{acc}}} \tag{3.2}$$

The total capacity loss is the sum $Q_{lost} = d_{cal} + d_{cyc}$. This integrated model allows for the direct quantification of how different control strategies impact the battery's long-term State of Health (SoH), enabling the training of agents that balance profitability with battery preservation. A key feature of this framework is that the physical parameters for this model ($\epsilon_0, \epsilon_1, \epsilon_2, \zeta_0, \zeta_1, Q_{acc}$) are not fixed. They can be empirically calibrated from real-world experimental data using the provided `Fit_battery.py` script, as detailed in Section 3.5.

3.3 A Unified Experimentation and Evaluation Workflow

A key contribution of this thesis is the development of a unified and powerful experimentation workflow, orchestrated by the main script `run_experiments.py`. This script replaces the previous fragmented approach, providing a single, interactive interface to manage the entire lifecycle of training, benchmarking, and evaluation for V2G control agents. This workflow is designed to be both flexible for research and rigorous for evaluation, supporting the dual goals of developing specialized and generalized agents.

3.3.1 Orchestration via `run_experiments.py`

The `run_experiments.py` script acts as the central hub for all experimentation. It guides the user through an interactive command-line process, ensuring consistency and reproducibility. The key steps are:

1. **Algorithm Selection:** The user can select from a predefined list of algorithms to benchmark. This includes Deep Reinforcement Learning agents (e.g., SAC, DDPG+PER, TQC), classical optimization methods (Model Predictive Control), and rule-based heuristics (e.g., Charge As Fast As Possible).

[4] 34.

2. **Scenario Selection:** The script automatically detects all available `.yaml` configuration files, allowing the user to choose one or more scenarios for the experiment. This choice determines the mode of operation (single-domain vs. multi-scenario).

3. **Reward Function Selection:** The framework's flexibility is enhanced by allowing the user to dynamically select the reward function for the RL agents from the `reward.py` module.

4. **Training and Benchmarking:** Based on the user's selections, the script proceeds to the optional training phase and then to a comprehensive benchmark, saving all results in a timestamped directory.

3.3.2 Dual-Mode Training: Specialists and Generalists

The new workflow elegantly unifies the training of both "specialist" and "generalist" agents, a concept previously handled by separate scripts. The behavior is determined implicitly by the number of selected scenarios:

- **Single-Domain Specialization:** If the user selects a single scenario, the script trains an RL agent exclusively on that environment. This produces a specialist agent, optimized to extract maximum performance from a specific, known set of conditions (e.g., a particular charging station topology and price profile).

- **Multi-Scenario Generalization:** If multiple scenarios are selected, the script automatically utilizes the `MultiScenarioEnv` wrapper. This custom Gymnasium environment dynamically switches between the different selected configurations at the start of each training episode. This process forces the agent to learn a robust and generalizable policy that performs well across a wide range of conditions, preventing overfitting to any single scenario. To handle the technical challenge of varying observation and action space sizes across scenarios, a `CompatibilityWrapper` is used to pad and slice the state-action vectors, enabling a single neural network policy to control heterogeneous environments.

3.3.3 Reproducible Benchmarking and Evaluation

To ensure a fair and scientifically valid comparison, the `run_benchmark` function implements a rigorous evaluation protocol. For each scenario, it first generates a "replay" file containing the exact sequence of stochastic events (e.g., EV arrivals, energy demands). This exact same sequence is then used to evaluate every algorithm, eliminating randomness as a factor in performance differences. The script runs multiple simulations for statistical robustness, aggregates the mean results, and automatically generates a suite of comparative plots, including overall performance metrics and detailed battery degradation analyses.

3.3.4 Interactive Web-Based Dashboard

To complement the command-line-driven workflow, the project includes an interactive web-based dashboard built with the Streamlit library, executed via the `streamlit_app.py` script. This graphical user interface (GUI) serves two primary functions, significantly enhancing usability and accessibility for experimentation and results analysis.

Simulation Orchestrator

The first part of the dashboard acts as a GUI wrapper for the `run_experiments.py` script. It provides a user-friendly web form where users can:

- Select which algorithms to benchmark from a multi-select list.
- Choose one or more scenarios to test.
- Pick a specific reward function for the RL agents from a dropdown menu.
- Set simulation parameters, such as the number of evaluation runs.
- Toggle optional steps, like running the `Fit_battery.py` calibration or enabling RL model training.

Upon clicking the "Run Simulation" button, the application constructs the equivalent command-line arguments and executes `run_experiments.py` as a subprocess. It captures and displays the console output in real-time on the web page, providing a seamless user experience without requiring direct terminal interaction.

Results Visualizer

The second part of the dashboard is a dedicated results browser. It automatically scans the `results/` directory and presents a list of all completed benchmark runs (organized by timestamp). The user can select a specific benchmark, and the application will find and display all the generated plots (e.g., performance comparisons, battery degradation graphs) directly on the page. This feature allows for quick and convenient inspection and comparison of outcomes from different experiments.

3.4 Evaluation Metrics

To ensure a fair and comprehensive comparison, all algorithms are evaluated against an identical set of pre-generated scenarios through a "replay" mechanism. The **mean** and **standard deviation** of performance are calculated across multiple simulation runs. The key metrics include:

- **Total Profit ($):** The net economic outcome, calculated as revenue from energy sales minus the cost of energy purchases.

$$\Pi_{\text{total}} = \sum_{t=0}^{T_{\text{sim}}} \sum_{i=1}^{N} \left(C_{\text{sell}}(t) P_{\text{dis},i}(t) - C_{\text{buy}}(t) P_{\text{ch},i}(t) \right) \Delta t$$

- **Tracking Error (RMSE, kW):** For grid-balancing scenarios, this measures the root-mean-square error between the fleet's aggregated power and a target setpoint.

$$E_{\text{track}} = \sqrt{\frac{1}{T_{\text{sim}}} \sum_{t=0}^{T_{\text{sim}}-1} \left(P_{\text{setpoint}}(t) - P_{\text{total}}(t) \right)^2}$$

- **User Satisfaction (Average):** The fraction of energy delivered compared to what was requested by the user, averaged across all EV sessions. A score of 1 indicates perfect service.

$$US_{\text{avg}} = \frac{1}{N_{\text{EVs}}} \sum_{k=1}^{N_{\text{EVs}}} \min \left(1, \frac{E_k(t_k^{\text{dep}})}{E_k^{\text{des}}} \right)$$

- **Transformer Overload (kWh):** The total energy that exceeded the transformer's rated power limit. An ideal controller should achieve a value of 0.

$$O_{\text{tr}} = \sum_{t=0}^{T_{\text{sim}}} \sum_{j=1}^{N_T} \max(0, P_j^{\text{tr}}(t) - P_j^{\text{tr,max}}) \cdot \Delta t$$

- **Battery Degradation ($):** The estimated monetary cost of battery aging due to both cyclic and calendar effects.

$$D_{\text{batt}} = \sum_{k=1}^{N_{\text{EVs}}} (\text{CyclicCost}_k + \text{CalendarCost}_k)$$

3.5 Simulator Implementation Details

During the analysis and implementation of new metrics, fundamental details about the `EV2Gym` simulator's architecture emerged, which warrant documentation. The configuration of Electric Vehicles (EVs) and the calculation of their degradation follow a specific logic dependent on a key parameter in the `.yaml` configuration files.

Vehicle Definition Modes

The simulator operates in two distinct modes, controlled by the boolean flag `heterogeneous_ev_specs`:

- **Heterogeneous Mode (`True`):** In this mode, the simulator ignores the default vehicle specifications in the `.yaml` file. Instead, it loads a list of vehicle profiles from an external JSON file, specified by the `ev_specs_file` parameter
(e.g., `ev_specs_v2g_enabled2024.json`). This allows for the creation of a realistic fleet with diverse battery capacities, charging powers, and efficiencies. For instance, the fleet may include a **Peugeot 208** with a 46.3 kWh battery and a 7.4 kW charge rate, alongside a **Volkswagen ID.4** with a 77 kWh battery and an 11 kW charge rate. A vehicle is randomly selected from this list for each new arrival event.

- **Homogeneous Mode (`False`):** In this mode, the external JSON file is ignored. All vehicles created in the simulation are identical, and their characteristics are defined exclusively by the `ev:` block within the `.yaml` configuration file. The `battery_capacity` parameter in this block becomes the single source of truth for the entire fleet.

Empirical Calibration of the Degradation Model

A significant enhancement in this work is the move towards a more physically representative and flexible battery degradation model. While the underlying semi-empirical model for calendar and cyclic aging remains, the methodology for parameterizing it has been fundamentally improved, addressing previous inconsistencies. This is achieved through the `Fit_battery.py` script, a new utility for empirical model calibration. The script implements the following workflow:

1. **Data Loading:** It loads time-series data from real-world battery aging experiments. The expected data includes measurements of capacity loss over time, along with contextual variables like state of charge (SoC), temperature, and energy throughput.

2. **Model Fitting:** Using the `curve_fit` function from the SciPy library, the script fits the parameters of the `Qlost_model` (which combines calendar and cyclic aging) to the empirical data. This optimization process finds the physical constants (e.g., ϵ_0, ζ_0) that best explain the observed degradation.

3. **Parameter Export:** The script outputs the calibrated parameters. These values can then be used directly in the simulator's configuration, ensuring that the degradation model for a specific EV fleet is grounded in experimental evidence for that battery type.

This calibration workflow, integrated optionally into the main `run_experiments.py` script, elevates the simulation's fidelity. It allows the framework to move beyond a single, fixed degradation model (previously calibrated for a 78 kWh battery) and enables the creation of high-fidelity digital twins for a wide variety of EV batteries, provided that the necessary experimental data is available.

3.6 Reinforcement Learning Formulation

The control problem is formalized as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ).

3.6.1 State Space (S)

The state $s_t \in S$ is a feature vector providing a snapshot of the environment at time t. A representative state, as defined in modules like `V2G_profit_max_loads.py`, includes:

$$s_t = [t, P_{\text{total}}(t-1), \mathbf{c}(t,H), \mathbf{L}_1(t,H), \mathbf{PV}_1(t,H), \dots, \mathbf{s}_1^{\text{EV}}(t), \dots, \mathbf{s}_N^{\text{EV}}(t)]^T$$

where the components are:

- t: The current time step.
- $P_{\text{total}}(t-1)$: The aggregated power from the previous time step.
- $\mathbf{c}(t,H)$: A vector of **predicted future** electricity prices over a horizon H.
- $\mathbf{L}_j(t,H), \mathbf{PV}_j(t,H)$: Forecasts for inflexible loads and solar generation.
- $\mathbf{s}_i^{\text{EV}}(t) = [\text{SoC}_i(t), t_i^{\text{dep}} - t]$: Key information for each EV i, including its State of Charge and remaining time until departure.

3.6.2 Action Space (A)

The action $a_t \in A$ is a continuous vector in $\mathbb{R}^N$, where N is the number of chargers. For each charger i, the command $a_i(t) \in [-1, 1]$ is a normalized value that is translated into a power command:

- If $a_i(t) > 0$, the EV is charging: $P_i(t) = a_i(t) \cdot P_{\text{charge},i}^{\max}$.
- If $a_i(t) < 0$, the EV is discharging (V2G): $P_i(t) = a_i(t) \cdot P_{\text{discharge},i}^{\max}$.

3.6.3 Reward Function

The reward function $R(t)$ encodes the objectives of the control agent. The framework allows for the selection of different reward functions from the `reward.py` module to suit various goals. Key examples include:

- **Profit Maximization with Penalties** (`ProfitMax_TrPenalty_UserIncentives`): This function creates a balance between economic gain and physical constraints.

$$R(t) = \underbrace{\text{Profit}(t)}_{\text{Economic Gain}} - \underbrace{\lambda_1 \cdot \text{Overload}(t)}_{\text{Grid Penalty}} - \underbrace{\lambda_2 \cdot \text{Unsatisfaction}(t)}_{\text{User Penalty}}$$

  The agent is rewarded for profit but penalized for overloading transformers and for failing to meet the charging needs of departing drivers.

- **Squared Tracking Error** (`SquaredTrackingErrorReward`): Used for grid service applications where precision is paramount.

$$R(t) = -\left(P_{\text{setpoint}}(t) - \sum_{i=1}^{N} P_i(t)\right)^2$$

  The reward is the negative squared error from the power setpoint, incentivizing the agent to minimize this error at all times.

By using this enhanced framework, this thesis moves beyond single-scenario optimization to develop and validate an intelligent V2G control agent that is not only high-performing but also robust, adaptable, and ready for the complexities of real-world deployment.

3.7 Reinforcement Learning Algorithms

This work benchmarks several state-of-the-art Deep Reinforcement Learning algorithms. The following sections provide a detailed mathematical description of the selected off-policy, actor-critic algorithms.

Soft Actor-Critic (SAC)

SAC is an off-policy actor-critic algorithm designed for continuous action spaces that optimizes a stochastic policy. Its core feature is entropy maximization, which encourages exploration and improves robustness. The agent aims to maximize not only the expected sum of rewards but also the entropy of its policy.

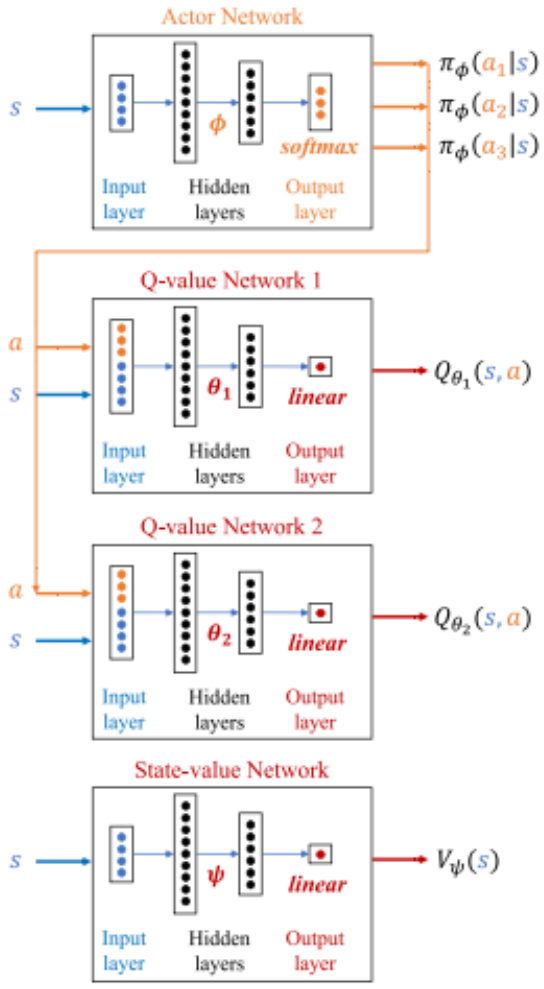


Figure 3.2: SAC Structure Image from[5]

Soft Actor-Critic (SAC) The objective function is:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right]$$

where $\mathcal{H}$ is the entropy of the policy π and α is the temperature parameter, which controls the trade-off between reward and entropy.

Implementation Details The implementation of SAC is built upon the robust, industry-standard **Stable-Baselines3** library, which provides a highly optimized and well-tested version of the algorithm. The standard `SAC` class from this library is used directly, leveraging its PyTorch-based backend for efficient training and inference.

SAC uses a soft Q-function, trained to minimize the soft Bellman residual:

$$L(\theta_Q) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim D} \left[\left(Q(s_t, a_t) - \left(r_t + \gamma V_{\bar{\psi}}(s_{t+1}) \right) \right)^2 \right]$$

where D is the replay buffer and the soft state value function V is defined as:

$$V_{\text{soft}}(s_t) = \mathbb{E}_{a_t \sim \pi}[Q_{\text{soft}}(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$$

To mitigate positive bias, SAC employs two Q-networks (Clipped Double-Q) and takes the minimum of the two target Q-values during the Bellman update.

Deep Deterministic Policy Gradient + PER (DDPG+PER)

DDPG is an off-policy algorithm that concurrently learns a deterministic policy $\mu(s|\theta^\mu)$ and a Q-function $Q(s, a|\theta^Q)$. It is the deep-learning extension of the DPG algorithm for continuous action spaces.

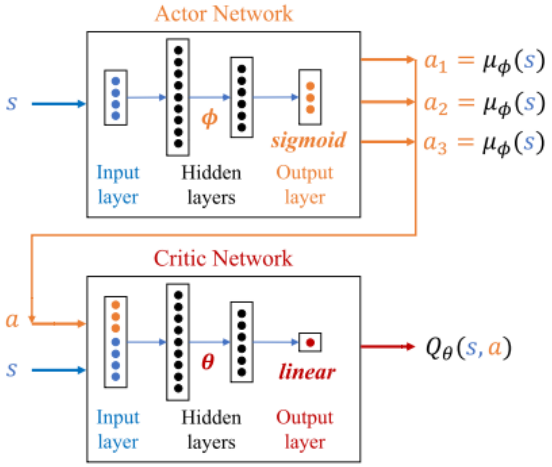


Figure 3.3: DDPG Structure (Image from[6]

- **Critic Update:** The critic is updated by minimizing the mean-squared Bellman error, similar to Q-learning. Target networks (Q' and μ') are used to stabilize training.

  $$L(\theta^Q) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim D}\left[(y_t - Q(s_t, a_t|\theta^Q))^2\right]$$

  where the target y_t is given by:

  $$y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'})$$

- **Actor Update:** The actor is updated using the deterministic policy gradient theorem:

  $$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim D}[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s_t|\theta^\mu)]$$

- **Prioritized Experience Replay (PER):** This work enhances DDPG with PER. Instead of uniform sampling from the replay buffer D, PER samples transitions (MILP) their TD-error, prioritizing those where the model has the most to learn. The probability of sampling transition i is:

  $$P(i) = \frac{p_i^\beta}{\sum_k p_k^\beta}$$

where $p_i = |\delta_i| + \epsilon$ is the priority based on the TD-error δ_i, and β controls the degree of prioritization. To correct for the bias introduced by non-uniform sampling, PER uses importance-sampling (IS) weights.

Implementation Details To integrate Prioritized Experience Replay (PER) with DDPG, a custom class, `CustomDDPG`, was developed in the `ev2gym/rl_agent/custom_algorithms.py` module. This class inherits from the standard `DDPG` agent provided by **Stable-Baselines3**. The core `train` method is overridden to replace the default uniform-sampling replay buffer with one that supports prioritized sampling. This involves calculating TD-errors for each transition, updating their priorities in the buffer, and using the resulting importance-sampling weights during the critic update, thereby focusing the learning process on the most informative experiences.

Truncated Quantile Critics (TQC)

TQC enhances the stability of SAC by modeling the entire distribution of returns instead of just its mean. This is achieved through quantile regression and a novel truncation mechanism to combat Q-value overestimation.

- **Distributional Learning:** TQC employs a set of N critic networks, $\{Q_{\phi_i}(s,a)\}_{i=1}^{N}$, each trained to estimate a specific quantile τ_i of the return distribution. The target quantiles are implicitly defined as $\tau_i = \frac{i-0.5}{N}$. The critics are trained by minimizing the quantile Huber loss, L_{QH}.
- **Distributional Target Calculation:** A distributional target is constructed for the Bellman update. First, an action is sampled from the target policy for the next state: $\tilde{a}_{t+1} \sim \pi_{\theta'}(\cdot|s_{t+1})$. Then, a set of N Q-value estimates for the next state is obtained from the N target critic networks: $\{Q_{\phi'_j}(s_{t+1}, \tilde{a}_{t+1})\}_{j=1}^{N}$.
- **Truncation:** This is the key idea of TQC. To combat overestimation, the algorithm discards the k largest Q-value estimates from the set of N target values. This truncation removes the most optimistic estimates, which are a primary source of bias, leading to more conservative and stable updates.
- **Critic Update:** The target value for updating the i-th critic is formed using the Bellman equation with the truncated set of next-state Q-values. The overall critic loss is the sum of the quantile losses across all critics:

$$L(\phi) = \sum_{i=1}^{N} \mathbb{E}_{(s,a,r,s')\sim D} \left[L_{QH} \left(r + \gamma Q_{\text{trunc}}(s', \tilde{a}') - Q_{\phi_i}(s,a) \right) \right]$$

where Q_{trunc} represents the value derived from the truncated set of target quantiles.

Implementation Details The TQC algorithm is leveraged from the **SB3-Contrib** library, a collection of community-contributed extensions to Stable-Baselines3. Using the library's standard `TQC` implementation allows the framework to benefit from this state-of-the-art distributional RL algorithm without requiring a custom implementation from scratch.

3.7.1 A History-Based Adaptive Reward for Profit Maximization

Implementation Details This reward function, along with a suite of other reward strategies, is implemented in the `ev2gym/rl_agent/reward.py` module. The main experimentation script,
Economic Profit

The foundation of the reward signal is the direct, instantaneous economic profit, Π_t. This component provides a clear and strong incentive for the agent to learn market dynamics, encouraging it to charge during low-price periods and discharge (V2G) during high-price periods.

$$\Pi_t = \sum_{i=1}^{N} \left(C_t^{\text{sell}} \cdot P_{i,t}^{\text{dis}} - C_t^{\text{buy}} \cdot P_{i,t}^{\text{ch}} \right) \Delta t \tag{3.3}$$

where N is the number of connected EVs, C_t^{sell} and C_t^{buy} are the electricity prices, and $P_{i,t}^{\text{dis}}$ and $P_{i,t}^{\text{ch}}$ are the discharging and charging powers for EV i.

Adaptive User Satisfaction Penalty

The penalty for failing to meet user charging demands, P_t^{sat}, is not a fixed value. Instead, it adapts based on the system's recent history of performance. The environment maintains a short-term memory of the average user satisfaction over the last 100 timesteps. From this history, we calculate an average satisfaction score, $\bar{S}_{hist}$.

A *satisfaction severity multiplier*, λ_t^{sat}, is then calculated. This multiplier grows quadratically as the historical average satisfaction drops, meaning that if the system has been performing poorly, the consequences for a new failure become much more severe.

$$\lambda_t^{\text{sat}} = \lambda_{\text{base}}^{\text{sat}} \cdot (1 - \bar{S}_{hist})^2 \tag{3.4}$$

where $\lambda_{\text{base}}^{\text{sat}}$ is a base scaling factor (e.g., 20.0). A penalty is only applied if any departing EV's satisfaction, S_k, is below a critical threshold (e.g., 95%). The magnitude of the penalty is the product of the adaptive multiplier and the current satisfaction deficit.

$$P_t^{\text{sat}} = \lambda_t^{\text{sat}} \cdot (1 - \min(S_k)) \quad \forall k \in \text{EVs departing at } t \tag{3.5}$$

This creates a powerful feedback loop: a single, isolated failure in an otherwise well-performing system results in a mild penalty. However, persistent failures lead to a rapidly escalating penalty, forcing the agent to correct its behavior.

Adaptive Transformer Overload Penalty

Similarly, the transformer overload penalty, P_t^{tr}, adapts based on the recent frequency of overloads. The environment tracks how often an overload has occurred in the last 100 timesteps, yielding an overload frequency, F_{hist}^{tr}. This frequency is used to compute a linear *overload severity multiplier*, λ_t^{tr}. The more frequently overloads have happened, the higher the penalty for a new one.

$$\lambda_t^{\text{tr}} = \lambda_{\text{base}}^{\text{tr}} \cdot F_{hist}^{\text{tr}} \tag{3.6}$$

where $\lambda_{\text{base}}^{\text{tr}}$ is a base scaler (e.g., 50.0). If the total power drawn, $P_j^{\text{total}}(t)$, exceeds the transformer's limit, $P_j^{\max}$, a penalty is applied. This penalty consists of a small, fixed base amount plus the adaptive component, which scales with the magnitude of the current overload.

$$P_t^{\text{tr}} = P_{\text{base}} + \lambda_t^{\text{tr}} \cdot \sum_{j=1}^{N_T} \max(0, P_j^{\text{total}}(t) - P_j^{\max}) \tag{3.7}$$

This mechanism teaches the agent that while a rare, minor overload might be acceptable in pursuit of high profit, habitual overloading is an unsustainable and heavily penalized strategy.

Rationale and Significance

This history-based adaptive reward function represents a significant advancement over static or purely state-based approaches. By making the penalty weights a function of the system's recent performance history, we provide a more nuanced and stable learning signal. The agent is not punished excessively for isolated, exploratory actions that might lead to a minor constraint violation. Instead, it is strongly discouraged from developing policies that lead to chronic system failures. The intuition is to mimic a more realistic management objective: maintain high performance on average, and react strongly only when performance trends begin to degrade. This method is also computationally efficient, avoiding complex state-dependent calculations in favor of simple updates to historical data queues. Ultimately, this reward structure guides the agent to discover policies that are not only profitable but also robust and reliable over time, striking a more intelligent balance between economic ambition and operational safety.

3.8 Online MPC Formulation (PuLP Implementation)

The Model Predictive Control (MPC) implemented with PuLP solves a profit maximization problem at each time step t over a finite prediction horizon H. This formulation is designed for online, real-time control, where decisions are made based on the current system state and future predictions.

Implementation Details This online controller is implemented as the `OnlineMPC_Solver` class within the `ev2gym/baselines/pulp_mpc.py` module. At each invocation of its `get_action` method, it dynamically constructs the full Mixed-Integer Linear Program (MILP) described below using the **PuLP** modeling library. The problem is then solved using the default CBC (COIN-OR Branch and Cut) solver.

3.8.1 Mathematical Formulation

At each time step t, the MPC controller solves the following optimization problem.

Objective Function: Net Operational Profit

The objective is to maximize the total net operational profit over the control horizon H. This provides a comprehensive economic model that goes beyond simple energy arbitrage.

$$\max_{P^{\text{ch}}, P^{\text{dis}}, z} \sum_{k=t}^{t+H-1} \sum_{i \in \text{CS}} \left(\text{Revenues}_{i,k} - \text{Costs}_{i,k} \right) \tag{3.8}$$

The revenue and cost components are defined for each station i at time step k as:

- **Revenues** consist of:
  - Grid Sales Revenue (V2G): $c_k^{\text{sell}} \cdot P_{i,k}^{\text{dis}} \cdot \Delta t$
  - User Charging Revenue: $c^{\text{user}} \cdot P_{i,k}^{\text{ch}} \cdot \Delta t$
- **Costs** consist of:
  - Grid Purchase Cost: $c_k^{\text{buy}} \cdot P_{i,k}^{\text{ch}} \cdot \Delta t$
  - Battery Degradation Cost: $c^{\text{deg}} \cdot (P_{i,k}^{\text{ch}} + P_{i,k}^{\text{dis}}) \cdot \Delta t$

where c_k^{sell} and c_k^{buy} are the time-varying electricity prices, c^{user} is the fixed price for the end-user, c^{deg} is the estimated cost of battery degradation per kWh cycled, and Δt is the time step duration.

System Constraints

The optimization is subject to the following constraints for each station i and time step $k \in [t, t + H - 1]$.

Energy Balance Dynamics. The state of energy of the EV battery evolves according to:

$$E_{i,k} = E_{i,k-1} + \left(\eta_{\text{ch}} P_{i,k}^{\text{ch}} - \frac{1}{\eta_{\text{dis}}} P_{i,k}^{\text{dis}} \right) \cdot \Delta t \tag{3.9}$$

where the initial state $E_{i,t-1}$ is the currently measured energy level of the EV.

Power Limits and Mutual Exclusion. Charging and discharging powers are bounded by the EV's capabilities and controlled by a binary variable $z_{i,k}$ to prevent simultaneous operation.

$$0 \leq P_{i,k}^{\text{ch}} \leq P_i^{\text{ch,max}} \cdot z_{i,k} \tag{3.10}$$

$$0 \leq P_{i,k}^{\text{dis}} \leq P_i^{\text{dis,max}} \cdot (1 - z_{i,k}) \tag{3.11}$$

State of Energy (SoE) Limits. The battery energy level must remain within its physical operational window.

$$E_i^{\min} \leq E_{i,k} \leq E_i^{\max} \tag{3.12}$$

User Satisfaction (Hard Constraint). The desired energy level must be met at the time of departure. This is modeled as a hard constraint, reflecting a non-negotiable service requirement.

$$E_{i,k_{\text{dep}}} \geq E_i^{\text{des}} \tag{3.13}$$

where k_{dep} is the predicted departure step of the EV within the horizon.

Transformer Power Limit. The total net power drawn from (or injected into) the grid by all charging stations must not exceed the transformer's maximum capacity.

$$\sum_{i \in \text{CS}} (P_{i,k}^{\text{ch}} - P_{i,k}^{\text{dis}}) \leq P^{\text{tr,max}} \tag{3.14}$$

Problem Classification

In the field of Operations Research, a Mixed-Integer Linear Program (MILP) is a powerful modeling tool for optimization problems involving complex decisions. A problem is classified as a MILP if it seeks to optimize a linear objective function, subject to a set of linear equality and inequality constraints, where the decision variables can be a mix of continuous and integer values. The general mathematical formulation of a MILP can be expressed as follows:

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & c^T x + h^T y \\ \text{subject to} \quad & Ax + Gy \leq b \\ & x \in \mathbb{Z}^n, \quad y \in \mathbb{R}^p \end{aligned} \tag{3.15}$$

where x represents the vector of integer variables and y represents the vector of continuous variables. The vectors c and h contain the objective function coefficients, while A, G, and b define the linear constraints of the system. The requirement for some variables to be integers ($x \in \mathbb{Z}^n$) is what makes MILPs fundamentally different from standard Linear Programs (LPs) and significantly more challenging to solve, forming a core part of the field of combinatorial optimization. Based on this definition, the mathematical structure of the optimization

problem described above is a classic **Mixed-Integer Linear Program (MILP)**. This classification is justified as follows:

- **Linear Objective Function:** The objective function is a linear combination of the continuous power variables P^{ch} and P^{dis}.
- **Linear Constraints:** All system constraints, including energy dynamics, power limits, and state of energy bounds, are formulated as linear equations or inequalities.
- **Mixed-Integer Variables:** The formulation employs both continuous variables (e.g., $P^{\text{ch}}_{i,k}$, $E_{i,k}$) and discrete, binary integer variables ($z_{i,k}$). The binary variables are essential for modeling the logical decision to either charge or discharge at any given time step, but not both simultaneously.

The problem is not a Quadratic Program (QP) or Mixed-Integer Quadratic Program (MIQP) because the objective function does not contain any quadratic terms (e.g., minimizing the square of power). Similarly, it is not a Quadratically Constrained Quadratic Program (QCQP) as all constraints are linear. The `OnlineMPC_Solver` is therefore designed to solve this specific MILP formulation at each control step.

3.9 Lyapunov-based Adaptive Horizon MPC

While the A-MPC offers a significant speed-up, it is an approximation and may not always match the performance of the fully-fledged online MPC. A second enhancement developed in this work is the **Lyapunov-based Adaptive Horizon MPC**, which aims to reduce the computational burden of the online MPC while retaining its optimality and stability guarantees. This method represents an essential improvement, creating an intelligent trade-off between computational cost and control performance.

Implementation Details This adaptive horizon mechanism is not a separate controller but rather an advanced operational mode of the `OnlineMPC_Solver` class. Its logic is integrated directly within the solver's `get_action` method and is activated by setting the `use_adaptive_horizon` flag to true during instantiation. When active, the solver performs the Lyapunov stability check after each optimization step and adjusts its internal horizon parameter, `current_H`, accordingly for the next iteration.

3.9.1 Core Concept: Dynamic Horizon Adjustment

The key insight is that a long prediction horizon is not always necessary. When the system is in a stable state and far from its operational constraints, a shorter horizon is sufficient for making good decisions. Conversely, when the system is in a complex or critical state (e.g., an EV is close to its departure time but has a low SoC), a longer horizon is needed for careful planning. This adaptive controller

dynamically adjusts its prediction horizon H_t at each step based on the stability of the system, which is formally assessed using a Lyapunov function.

3.9.2 Lyapunov Stability for V2G Control

A Lyapunov function $V(x)$ is a scalar function that can be thought of as a measure of the system's "energy" or deviation from a desired equilibrium state. For the V2G system, we define the state as the vector of energy levels of all connected EVs, $E = [E_1, E_2, \ldots, E_N]^T$. The desired state is the vector of desired energy levels at departure, E^{des}. The Lyapunov function is defined as the sum of the squared errors from this desired state:

$$V(E) = \sum_{i \in \text{EVs}} (E_i - E_i^{\text{des}})^2 \tag{3.16}$$

For the system to be stable, the value of this function must decrease at each step, ensuring the system is always progressing towards its goal. This is known as the Lyapunov decrease condition:

$$V(E_{t+1}) \leq V(E_t) - \alpha V(E_t) \tag{3.17}$$

where E_{t+1} is the state at the next time step resulting from the current control action, and α is a small positive constant that sets the minimum required rate of convergence.

3.9.3 Horizon Shortening and Extension

The adaptive MPC algorithm uses this stability condition to govern its horizon length. At each time step t:

1. The MPC solves the optimization problem using its current horizon, H_t.
2. It calculates the predicted next state E_{t+1} based on the computed optimal action.
3. It checks if the Lyapunov decrease condition is satisfied.
    - **If Stable:** The condition holds. The controller is performing well. We can afford to reduce the computational load for the next step by shortening the horizon:
    $$H_{t+1} = \max(H_{\min}, H_t - 1) \tag{3.18}$$
    - **If Not Stable:** The condition is violated. The system requires more careful planning. The horizon for the next step is extended to provide a longer view into the future:
    $$H_{t+1} = \min(H_{\max}, H_t + 1) \tag{3.19}$$

This intelligent adjustment makes the online MPC more efficient and practical, reducing computation time during stable periods while retaining the ability to perform deep planning when necessary.

3.10 Approximate Explicit MPC: A Machine Learning Approach

The online, implicit MPC formulation provides high-quality control decisions by solving an optimization problem at every time step. However, this approach has a significant drawback: its computational complexity. For scenarios with a large number of EVs or a long control horizon, solving a Mixed-Integer Linear Program (MILP) in real-time can be prohibitively slow, making it impractical for many real-world applications.

To overcome this limitation, this work implements an **Approximate Explicit Model Predictive Controller (A-MPC)**. This controller leverages machine learning to replace the computationally expensive online optimization with a fast, lightweight inference step.

Implementation Details The A-MPC is implemented in the `ApproximateExplicitMPC` class, located in the same `ev2gym/baselines/pulp_mpc.py` file. It utilizes a pre-trained `RandomForestRegressor` model from the **scikit-learn** library. This model is serialized and stored in the `mpc_approximator.joblib` file, and it is generated by the dedicated `train_mpc_approximator.py` script, which executes the data generation and offline training steps. During online operation, the controller's `get_action` method simply performs a fast inference call to this loaded model.

3.10.1 Methodology: From Oracle to Apprentice

The core idea is to treat the slow but powerful online MPC as an "oracle" or expert teacher. An apprentice model, in this case a `RandomForestRegressor` from the `scikit-learn` library, is trained to mimic the oracle's behavior. The process is as follows:

1. **Data Generation:** The online MPC is run across a diverse range of simulated scenarios. At each step, the state of the environment and the corresponding optimal action computed by the MPC are recorded. This creates a large dataset of state-action pairs, where the actions are considered to be the "ground truth" optimal decisions.

2. **State Vector Formulation:** The state s_t fed to the machine learning model is a carefully crafted vector that summarizes all necessary information for making a control decision. It is a fixed-size vector composed of:

$$s_t = [\mathbf{SoC}, \mathbf{T}^{\text{rem}}, \mathbf{C}^{\text{ch}}, \mathbf{C}^{\text{dis}}]^T \tag{3.20}$$

   where:

   - **SoC**: A vector of the current State of Charge for all charging stations (padded to a maximum size).
   - $\mathbf{T}^{\text{rem}}$: A vector of the remaining time until departure for each connected EV.

- $\mathbf{C}^{\text{ch}}$: A vector of predicted future charging prices over the horizon H.
- $\mathbf{C}^{\text{dis}}$: A vector of predicted future discharging prices over the horizon H.

3. **Offline Training:** The `RandomForestRegressor` model, denoted f_θ, is trained offline on this dataset to learn the mapping from a given state s_t to the oracle's action a_t. The model's parameters θ are optimized to minimize the difference between its predicted action and the oracle's action.

4. **Online Inference:** Once trained, the A-MPC controller can be deployed. At each time step, it simply constructs the state vector s_t and computes the action via a fast forward pass through the trained model:

$$a_t = f_\theta(s_t) \tag{3.21}$$

This inference step is orders of magnitude faster than solving a MILP, enabling real-time control for large-scale systems.

3.10.2 A More Principled Approximator: The Deep ReLU Network

While the Random Forest provides a powerful and general-purpose approximation, Chapter 2 introduces a more theoretically grounded approach for this specific problem: using a deep neural network with Rectified Linear Unit (ReLU) activation functions. This method is not just an approximation but is theoretically capable of **exactly representing** the Piecewise Affine (PWA) nature of the explicit MPC control law. This work implements this advanced alternative.

Implementation Details This controller is implemented as the `ApproximateExplicitMPC_NN` class. It utilizes a PyTorch-based neural network, whose architecture is defined in `MPCApproximatorNet`. The model is trained using the dedicated `train_mpc_approximator_nn.py` script, which also parallelizes the data generation process for efficiency. The resulting trained model is saved to `mpc_approximator_nn.pth`.

Theoretical Motivation As detailed in Section 2.3 of Chapter 2, the explicit solution to a linear MPC problem is a PWA function. The work by Karg et al.[7] demonstrated that a deep ReLU network can perfectly replicate such a function. This provides a significant advantage over general-purpose approximators:

- **Representational Efficiency:** Deep ReLU networks can represent an exponentially large number of linear regions with only a linear increase in the number of parameters. This directly counters the "curse of dimensionality" that plagues traditional explicit MPC, allowing a compact model to represent a highly complex control law.

[7] 22.

- **Theoretical Soundness:** Using a model that matches the underlying mathematical structure of the solution is more principled and has the potential to yield a more accurate and robust controller.

Methodology The training methodology mirrors that of the Random Forest version but is adapted for a neural network:

1. **Oracle Data Generation:** The `train_mpc_approximator_nn.py` script uses the fixed-horizon `OnlineMPC_Solver` as an oracle to generate a large dataset of state-action pairs. This process is parallelized across multiple CPU cores to drastically reduce the time required.

2. **Offline Supervised Learning:** A Multi-Layer Perceptron (MLP) with hidden layers [256, 128, 64] and ReLU activations is trained on this dataset. The network learns to map the state vector s_t to the optimal power values a_t by minimizing the Mean Squared Error (MSE) loss function using the Adam optimizer.

3. **Fast Online Inference:** During simulation, the `ApproximateExplicitMPC_NN` controller performs a single, rapid forward pass of the state vector through the trained network to obtain the control action, achieving the real-time performance required.

This implementation provides a direct, practical realization of the advanced explicit MPC representation discussed in the state-of-the-art review, creating a powerful baseline for comparison.

Bibliography

[1] Feyijimi Adegbohun et al. "A Review of Bidirectional Charging Grid Support Applications and Control". In: *Energies* 17.6 (2024), p. 1320. DOI: 10.3390/en17061320.

[2] Fayiz Alfaverh, Mouloud Denai, and Yichuang Sun. "Optimal vehicle-to-grid control for supplementary frequency regulation using deep reinforcement learning". In: *Applied Energy* 325 (2022), p. 119881. DOI: 10.1016/j.apenergy.2022.119881.

[3] Dou An, Feifei Cui, and Xun Kang. "Optimal scheduling for charging and discharging of electric vehicles based on deep reinforcement learning". In: *Frontiers in Energy Research* 11 (2023), p. 1273820. DOI: 10.3389/fenrg.2023.1273820.

[4] Alberto Bemporad. "Explicit Model Predictive Control". In: *Springer Handbook of Automation* (2013), pp. 883–898.

[5] Alberto Bemporad and Carlo Filippi. "Suboptimal Explicit MPC via Approximate Multiparametric Quadratic Programming". In: *Proceedings of the 40th IEEE Conference on Decision and Control* (Dec. 2001), pp. 4851–4856.

[6] Christoph R. Birkl et al. "Degradation diagnostics for lithium ion cells". In: *Journal of Power Sources* 341 (2017), pp. 373–386.

[7] Francesca Bisetto. "La Duck Curve e la scomposizione della stagionalità dei consumi elettrici. Aspetti teorici e modelli statistici". Dipartimento di Scienze Statistiche, Correlatore: Luigi Grossi. Tesi di laurea magistrale. Padova: Università degli Studi di Padova, 2023.

[8] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[9] Greg Brockman et al. "OpenAI Gym". In: *arXiv preprint arXiv:1606.01540* (2016). DOI: 10.48550/arXiv.1606.01540.

[10] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer Science & Business Media, 2013.

[11] Viorica Rozina Chifu et al. "A Deep Q-Learning based Smart Scheduling of EVs for Demand Response in Smart Grids". In: *arXiv preprint arXiv:2401.02653* (2024). DOI: 10.48550/arXiv.2401.02653.

[12] European Commission. *Fit for 55: Delivering the EU's 2030 Climate Target*. Accessed: 2025-09-21. 2021. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A52021DC0550.

[13] C. R. Cutler and B. L. Ramaker. "Dynamic Matrix Control – A Computer Control Algorithm". In: *Proceedings of the Joint Automatic Control Conference*. Paper No. WP5-B. American Control Conference. San Francisco, USA, 1980.

[14] Gianluca Faggio. "Design and Testing of Online and Offline Optimization Algorithms for Vehicle-to-Grid (V2G) Industrial Applications". MA thesis. Politecnico di Milano, 2023.

[15] Kilian Freitag et al. "Curriculum Reinforcement Learning for Complex Reward Functions". In: *arXiv preprint arXiv:2410.16790* (2024). Emails: {tamino, cederk, laezza, knut.akesson, morteza.chehreghani}@chalmers.se.

[16] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018. URL: https://arxiv.org/abs/1802.09477.

[17] Shuifu Gu, Kejun Qian, and Yongbiao Yang. "Optimization of Electric Vehicle Charging and Discharging Strategies Considering Battery Health State: A Safe Reinforcement Learning Approach". In: *World Electric Vehicle Journal* 16.5 (2025), p. 286. DOI: 10.3390/wevj16050286.

[18] Tuomas Haarnoja et al. "Soft Actor-Critic Algorithms and Applications". In: *arXiv preprint arXiv:1812.05905* (2019). URL: https://arxiv.org/abs/1812.05905.

[19] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[20] Sinan Ibrahim et al. "Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications". In: *IEEE Access* PP.99 (2024). Accessed: 2025-09-21, pp. 1–1. DOI: 10.1109/ACCESS.2024.3504735. URL: https://arxiv.org/abs/2408.10215.

[21] J. L. TESTUD J. RICHALET A. RAULT and J. PAPON. "Model predictive heuristic control: Applications to industrial processes". In: *Automatica* 14.5 (1978), pp. 413–428. DOI: 10.1016/0005-1098(78)90001-8.

[22] Benjamin Karg and Sergio Lucia. "Efficient representation and approximation of model predictive control laws via deep learning". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3866–3878. DOI: 10.1109/TCYB.2020.2999556.

[23] Arthur J Krener. "Adaptive Horizon Model Predictive Control". In: *arXiv preprint arXiv:1602.08619* (2016). arXiv: 1602.08619 [math.OC].

[24] Arsenii Kuznetsov et al. "Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020. URL: https://proceedings.mlr.press/v119/kuznetsov20a/kuznetsov20a.pdf.

[25] Haijie Li et al. “Investigating Dynamic Behavior in SAG Mill Pebble Recycling Circuits: A Simulation Approach”. In: *Minerals* 14.7 (2024). Accessed: 2025-09-21, p. 716. DOI: 10.3390/min14070716. URL: https://www.mdpi.com/2075-163X/14/7/716.

[26] Timothy P Lillicrap et al. “CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING”. In: *arXiv preprint arXiv:1509.02971* (2015).

[27] Ding Liu et al. “Deep reinforcement learning for charging scheduling of electric vehicles considering distribution network voltage stability”. In: *Sensors* 23.3 (2023), p. 1618. DOI: 10.3390/s23031618.

[28] Horia Mania, Aurelia Guy, and Benjamin Recht. “Simple Random Search Provides a Competitive Approach to Reinforcement Learning”. In: *arXiv preprint arXiv:1803.07055* (2018). URL: https://arxiv.org/abs/1803.07055.

[29] D. Q. Mayne et al. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814. DOI: 10.1016/S0005-1098(99)00214-9.

[30] Carlos A Minchala-Ávila, Paúl Arévalo, and Diego Ochoa-Correa. “A Systematic Review of Model Predictive Control for Robust and Efficient Energy Management in Electric Vehicle Integration and V2G Applications”. In: *Modelling* 6.1 (2025), p. 20. DOI: 10.3390/modelling6010020.

[31] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016, pp. 1928–1937. URL: https://arxiv.org/abs/1602.01783.

[32] Andrew Y Ng, Daishi Harada, and Stuart Russell. “Policy invariance under reward transformations: Theory and application to reward shaping”. In: *ICML* 99 (1999), pp. 278–287.

[33] Bruce Nielson and Daniel C. Elton. “Induction, Popper, and Machine Learning”. In: *arXiv preprint* arXiv:2110.00840 (2021). cs.AI. eprint: 2110.00840. URL: https://arxiv.org/abs/2110.00840.

[34] Stylianos Orfanoudakis et al. “EV2Gym: A Flexible V2G Simulator for EV Smart Charging Research and Benchmarking”. In: *arXiv preprint arXiv:2404.01849* (2024). DOI: 10.48550/arXiv.2404.01849.

[35] Alessandra Parisio, Evangelos Rikos, and Luigi Glielmo. “A Model Predictive Control Approach to Microgrid Operation Optimization”. In: *IEEE Transactions on Control Systems Technology* 22.5 (2014), pp. 1813–1827. DOI: 10.1109/TCST.2013.2295737.

[36] Rey Pocius et al. “Comparing Reward Shaping, Visual Hints, and Curriculum Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2019). Oregon State University, School of EECS, Corvallis, OR 97331, USA | pociusr@oregonstate.edu; University of Pennsylvania, School of EAS, Philadelphia, Pennsylvania, 19104, USA | isele@seas.upenn.edu; Naval Research Laboratory, Code 5514; Washington, DC, 20375, USA.

[37] Dajun Qiu et al. "Reinforcement learning for vehicle-to-grid: A review". In: *Renewable and Sustainable Energy Reviews* 167 (2022), p. 112702.

[38] Dawei Qiu et al. "Reinforcement learning for electric vehicle applications in energy management: A review". In: *Renewable and Sustainable Energy Reviews* 163 (2023), p. 112443. DOI: 10.1016/j.rser.2022.112443.

[39] Mohammad Sadeghi. "Cost and power loss aware coalitions under uncertainty in transactive energy systems". In: *Université d'Ottawa / University of Ottawa* (2022). PhD Thesis.

[40] Gabriel Antonio Salvatti et al. "Electric Vehicles Energy Management with V2G/G2V Multifactor Optimization of Smart Grids". In: *Energies* 13.5 (2020), p. 1191. DOI: 10.3390/en13051191.

[41] Tom Schaul et al. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015). DOI: 10.48550/arXiv.1511.05952.

[42] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 3371–3380. URL: https://arxiv.org/abs/1707.06347.

[43] John Schulman et al. "Trust Region Policy Optimization". In: *Proceedings of the 32nd International Conference on Machine Learning* 37 (2015), pp. 1889–1897. URL: https://proceedings.mlr.press/v37/schulman15.html.

[44] G. Srihari et al. "Integration of electric vehicle into smart grid: a meta heuristic algorithm for energy management between V2G and G2V". In: *Frontiers in Energy Research* 12 (2024), p. 1357863. DOI: 10.3389/fenrg.2024.1357863.

[45] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second edition, in progress. A Bradford Book, c. 2014, 2015. Cambridge, Massachusetts; London, England: The MIT Press, 2015.

[46] Ahmad Tavakoli et al. "Impacts of grid integration of solar PV and electric vehicle on grid stability, power quality and energy economics: a review". In: *IET Energy Systems Integration* 2.3 (2020), pp. 233–245. DOI: 10.1049/iet-esi.2019.0047.

[47] Jürgen Vetter et al. "Ageing mechanisms in lithium-ion batteries". In: *Journal of Power Sources* 147.1-2 (2005), pp. 269–281.

[48] David Silver et. al Volodymyr Mnih Koray Kavukcuoglu. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[49] Mingyu Wang et al. "Multi-Agent Reinforcement Learning is a Sequence Modeling Problem". In: *arXiv preprint arXiv:2205.14953* (2022). URL: https://arxiv.org/abs/2205.14953.

[50] Zhaoyu Wang et al. "An electrical vehicle-assisted demand response management system: A reinforcement learning method". In: *Frontiers in Energy Research* 10 (2022), p. 1071948. DOI: 10.3389/fenrg.2022.1071948.

[51] Corey D. White and K. Max Zhang. "Using vehicle-to-grid technology for frequency regulation and peak-load reduction". In: *Journal of Power Sources* 196.3 (2011), pp. 3972–3980. DOI: 10.1016/j.jpowsour.2010.11.010.

[52] H. Xie. "Reinforcement learning for vehicle-to-grid: A review". In: *ScienceDirect* (2025). DOI: 10.1016/j.sciad.2025.100008.

[53] Na Xu et al. "A Review of Smart Grid Evolution and Reinforcement Learning: Applications, Challenges and Future Directions". In: *Energies* 18.7 (2024), p. 1837. DOI: 10.3390/en18071837.

[54] Yubao Zhang, Xin Chen, and Yuchen Zhang. "Transfer deep reinforcement learning-based large-scale V2G continuous charging coordination with renewable energy sources". In: *arXiv preprint arXiv:2210.07013* (2023).