POLITECNICO DI TORINO

MASTER OF SCIENCE THESIS

AUTOMOTIVE ENGINEERING

---

# Reinforcement Learning for Hybrid/electric vehicle: Analysis and performance of reward functions in a real-time algorithm for P2-HEV

---


**Advisors:**

Prof. Ezio Spessa

Prof. Daniela Anna Misul

Dott. Ing. Matteo Acquarone

Dott. Ing. Claudio Maino

**Candidate**

Pasquale Ciccullo



Academic year 2019/2020

# Abstract

Conventional vehicles with internal combustion engine (ICE) provide a good performance and long operating range by utilizing the high energy-density advantages of petroleum fuels. However, the conventional ICE-vehicles suffer the disadvantages of poor fuel economy and environmental air pollution. One of the immediate alternative solutions is the HEVs (Hybrid-Electric vehicle). In HEV, the introduction of one or more power sources increases the complexity of powertrain architecture and offers additional degrees of freedom in controlling the power-split between the power sources. In this energy management scenario, the Reinforcement learning (RL) allows to obtain a global optimization implemented in real-time differently from rule-based or optimization-based control strategies. In this work, a Deep Reinforcement Learning (DRL) algorithm, i.e., Double Deep-Q-network (DDQN), is adopted to control the power-split and the gear number. DDQN aims to simultaneously minimize the fuel consumption (FC) and maintain the state of charge (SOC) of the battery within the operating range. The case study is a parallel P2-HEV, passenger car. The software used is composed by three elements: *Simulator*, *Agent* and *Environment Interface*. The *Simulator*, is implemented in MATLAB, represents the model of the vehicle, and communicates with the *Agent* and *Environment Interface*, implemented in Python. The *Environment Interface* is the interface between the communication components of the *Agent* and physical simulation. The *Agent* has a logical interface divided into three elements: *Training algorithm*, *Approximator* (Artificial neural network) and *Exploration strategy* that is the component of the agent that allows to obtain optimal action-value function and find the optimal policy. The main goal of this work is to compare five different reward functions in order to demonstrate how this crucial function affects the performance of the algorithm. The DDQN algorithm and reward functions are analysed on four real driving cycles (CLUST), covering many possible vehicle driving scenarios. The results demonstrate how a good calibration on the reward coefficient allows to improve the effectiveness of the reward function, achieving a better performance and minimizing the fuel consumption (FC). The results are also compared to the Equivalent Consumption Minimization Strategy (ECMS) used as a benchmark energy management strategy.

# Contents

# List of figures

# List of tables

# 1. Introduction

## 1.1. Motivations

The automotive sector is in continuously changing and evolution in order to deal with air pollution (*figure 1*) and global warming mainly caused by CO2. Due to that, the EU legislative set the Climate and Energy package designed to reach three core targets by 2020 [1]:

- 20% reduction in EU greenhouse gas emissions from 1990 levels.
- 20% increase in the share of EU energy consumption produced from renewable resources.
- 20% improvement in the EU's energy efficiency.

In EU Road mobility and transport are globally asked to reduce more than 30% the TTW CO2 emissions in the next 10 years and more stringent regulation are incoming.

Since the BEV (Battery electric vehicle) solution cannot be the only solution able to satisfy these EU road mobility requirements since has some limitations related to the limited range (“range anxiety”), to the impact of battery cost on TCO (Total Cost of Ownership), to the thermal comfort (and related impact on driving range), to the mechanical protection of the battery, and to the charging time (customer acceptance).

In this scenario, the prediction is that in 2040/2050, the road transport show that efficiency and environmental targets will be reached by a suitable mix of technological solutions (improved efficiency, alternative and biofuels, electrification, hybrid solutions).

The HEVs and (P)HEVs represent a medium and long-term solution.

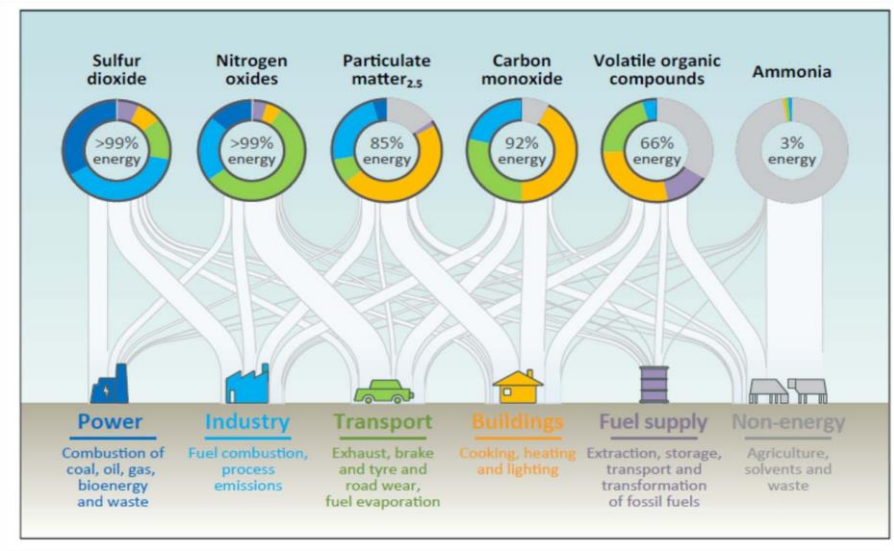


*Figure 1. Selected primary air pollutants and their sources*

## 1.2. Hybrid-electric vehicles (HEV)

### 1.2.1. Why Hybrid-electric vehicles (HEV)

In order to guarantee an alternative solution respect to the internal combustion engine vehicles (ICEs) that is an on-board energy-power source based on the chemical energy-power of a fuel (still today usually liquid hydrocarbons as gasoline or diesel); there are a lot of solutions available on the market like battery electric vehicles (BEVs), alternative/biofuels vehicles, fuel cell electric vehicles (FCEV) or electrification of the conventional powertrain (hybrid solution) [2].

The hybrid solution (HEVs) can be divided into three macro-categories (*figure 2*):

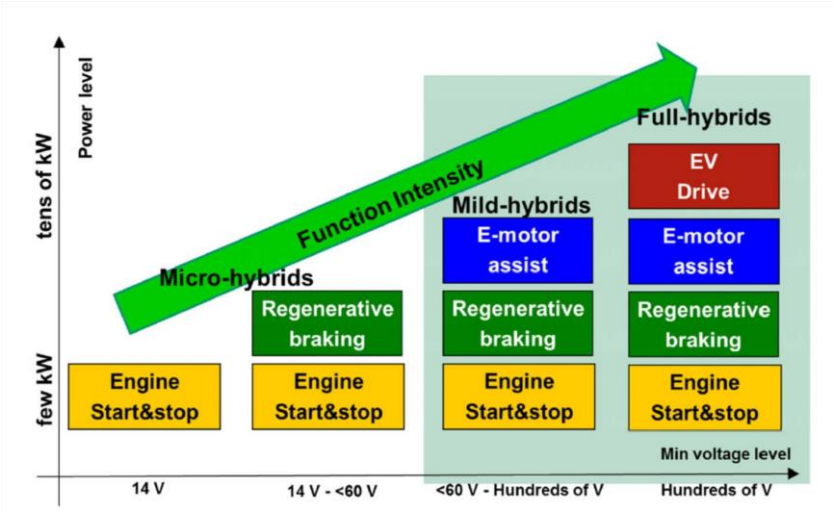


*Figure 2. HEVs classification based on power and voltage*

- **Micro-hybrid**: using the electric motor (EM) of few power [kW] as a belt driven starter generator (BSG) replacing the conventional alternator. The battery is a 12V battery (lithium-ion) that can be directly connected or indirectly connected to the 12V lead-acid battery through DC-DC converter. This solution guarantees a torque support during acceleration and ICE re-cranking (Start&Stop).
- **Mild-hybrid**: the battery is a 48V solution and 48V-12V DC-DC converter using e-machine (with power between 15 kW and 30 kW) in alternator position or between the transmission and engine. These solutions guarantee e-assist and e-boost (EM can support the ICE), downsizing the ICE, regenerative braking more effective, engine cranking more effective, small pure electric mode.
- **Full-Hybrid:** using high voltage battery (HV battery) with one or two e-machines with different positions (with a power between 10 kW and 100 kW). This solution is preferable because it guarantees all the features of an HEV in a more efficient way.

The hybrid solution is a perfect compromise between conventional vehicle (ICEs) and battery electric vehicles (BEVs). The characteristics of HEV are:

- **Regenerative braking**: in which part of the energy not needed to brake the vehicle is recovered into kinetic energy using the e-machine as an electric generator in order to recharge the battery.

- **ICE off in idle**: the usage of EM allows to avoid the ICE in idle avoiding the cold start and guaranteed the re-cranking of the ICE avoiding worst efficiency working points of the internal combustion engine causing HC/CO/NOx emissions. The pure electric transient motion can be used in less energy demanding request in short time like parking, queue conditions, reverse speed.

- **ICE better efficiency**: the e-motor guarantees e-assist (full performance acceleration using EM and ICE) and e-boost (EM torque boosting). These two features allow to properly work the ICE in more efficient working points.

- **ICE Downsizing/Downspeeding**: using an e-motor in the powertrain, the engine (ICE) can be sized with a low engine displacement (downsized) and it can work at low engine speed (downspeeding) guaranteeing better efficiency.

### 1.2.2. HEV powertrain architecture (HEV)

The HEV powertrain is composed by:

- **Internal combustion engine (ICE)**: the ICE used in HEVs solutions is downsized (1.0 L) rather than conventional vehicle in order to be complied with advanced technologies to reduce emissions and increase the efficiency of the powertrain, besides reduces weight. Typically, it is a gasoline engine since the coupling between diesel engine and electric motor is not efficient given the good performance of the diesel engine at low speed and low load and the air pollutants emitted by this type of engine, above all Nitrogen-oxide (NOx) and particulate matter (PM).
- **Fuel Tank**: in order to store the gasoline fuel.
- **Electric motor (EM)**: in HEVs solutions, it can be possible to have one or more electric motor on board depending on the solutions applied. Thanks to the DC/AC converter (inverter), the electric motor can act as a motor (to propel the vehicle) or as generator (to recharge the high voltage battery in braking).
- **Battery (B)**: lithium-ion high voltage battery with a energy stored between 1.5 to 10 kWh.
- **Multi-speed transmission**: allowing the mechanical energy coming from the ICE to be transmitted to the wheels.
- **Torque-coupling device**

There can be two possible HEV configurations: *Series-HEV* and *parallel-HEV*:

- **Simple Series-HEV**: the series hybrid propulsion system has only one powertrain with the electric motor as torque actuator. The hybridisation is realised at energy source level with one electric link (realised or directly or through a power converter) connecting one electric source (typically but not necessarily a battery pack) to an electric generation system based on an ICE mechanically coupled to an e-machine mainly or solely used as generator. Therefore, the electric transmission can be seen as a series hybrid with no electric source (*figure 3*).

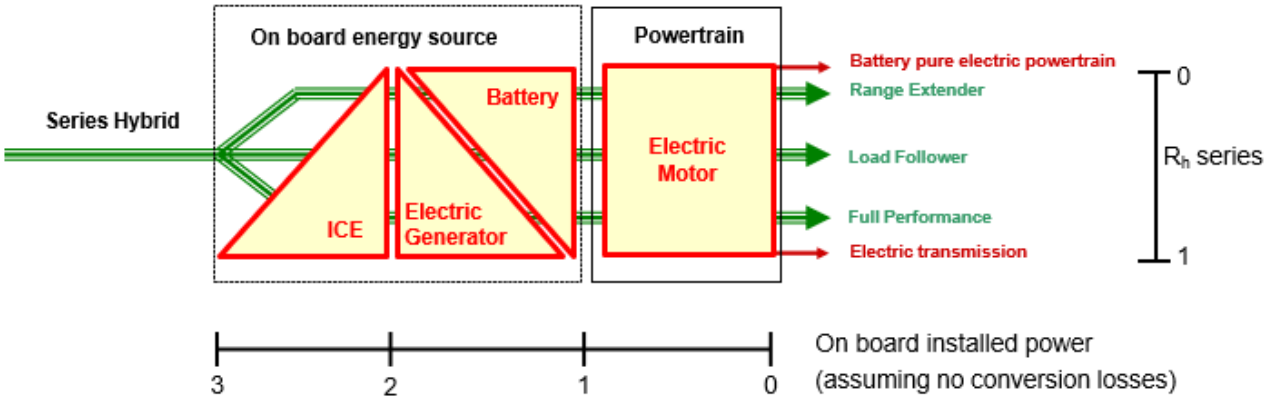


*Figure 3. Simple Thermal-Electric series-HEV*

In this type of configuration, it is possible to define series hybridization ratio $R_h series$ as the ration between the internal combustion engine power and the electric power:

$$R_h series = \frac{P_{ICE}}{P_{EM}}$$

Where:

- $R_h series = 1$: the vehicle is an electric transmission vehicle (series-HEV without battery).
- $R_h series = 0$: the vehicle is a battery electric vehicle (BEV).
- $R_h series = (0; 1)$: depending on the hybridisation ratio used in the sizing of the two electric power- energy sources, different series hybrid configurations can be defined:
    - *Range extender*: in which the power of hybridization unit (HU=ICE, e-generator and power converter) is equal to the average power of a reference cycle representative of the real usage.
    - *Load follower*: in which the HU power is equal to the continuous maximum power condition (load follower).
    - *Full performance*: in which the HU power is equal to to the transient maximum power condition.

- **Simple Parallel-HEV**: the simple parallel hybrid (*figure 4*) is a traction system made of two elementary traction systems (one based on an ICE and one on an e-motor) with as main energy source the ICE fuel stored in the on-board tank and as possible secondary energy source the electric energy of an electric storage system (typically a high voltage battery pack). The hybridisation is realised at powertrain level with a mechanical direct (through clutches, joints, gears) or an indirect link (with two powertrains one for each axle and coupled through the road). In order to limit the volume-weight oversizing and related extra-cost, usually the battery pack is sized for the power requests accepting to sacrifice the pure electric range (electrically assisted thermal engine-based vehicle)

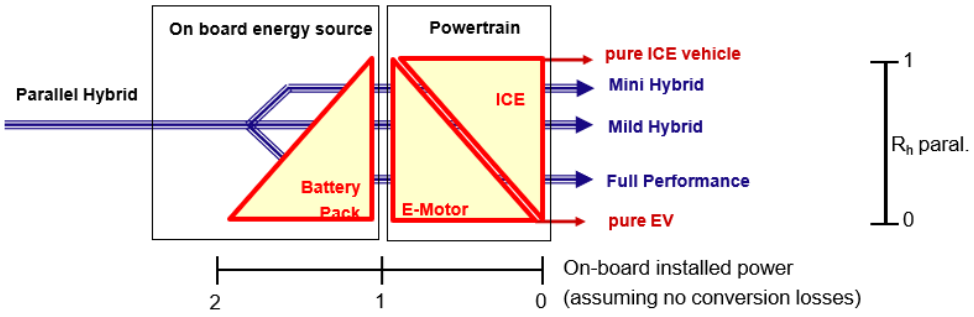


*Figure 4. Simple Thermal-electric parallel-HEV*

In this type of configuration, it is possible to define series hybridization ratio $R_h parallel$ as the ratio between the internal combustion engine power and the sum between internal combustion engine power and electric power:

$$R_h parallel = \frac{P_{ICE}}{P_{EM} + P_{ICE}}$$

Where:

- $R_h parallel = 1$: the vehicle is a pure conventional ICE-based vehicle.
- $R_h parallel = 0$: the vehicle is a pure electric vehicle (EV).
- $R_h parallel = (0; 1)$: depending on the hybridisation ratio used in the sizing of the two electric power-energy sources, different parallel hybrid configurations can be defined:
    - *Mini-Hybrid*: in which the electric motor is used as Belt stater generator (BSG) operating as starter to re-crank the ICE at each automatic cranking bypassing conventional starter motor. The high voltage battery is 12V.
    - *Mild-Hybrid*: the high voltage battery is at 48V. The electric motor is a 48 V solution in general with one e-machine (in the alternator position or between transmission and engine).
    - *Full performance (Full-hybrid)*: HV battery solution (in general hundreds of volts) with one (or two) e-machines (with different possible positions).

### 1.2.3. Parallel-HEV: Classifications based on e-machine position of parallel-HEV

There are two types of methods to classify the simple parallel-HEV:

- *Beretta method* (Traction method): it is a power/energy method used to evaluate the different traction systems from an energy perspective. It is based on the elementary traction system concept (sum of all the devices actively involved in the energy flux for the vehicle motion).
- *P-method*

- *Beretta method*

In Parallel Simple Thermal-Electric Hybrids (**Parallel-HEV**), the elementary traction systems (On-board energy/power source and powertrain) are connected **mechanically,** when the connection is at powertrain level. The two mechanical actuators are an Internal combustion engine (ICE) and an electric motor (EM)

The parallel-HEV are classified into (*figure 5*):

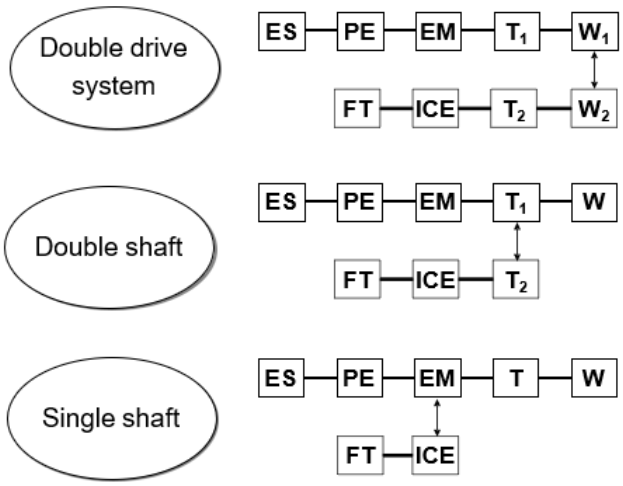


*Figure 5. Classification of simple parallel-HEV (Beretta method)*

- *Double drive system*: the connection between actuators is at wheel level, this is the typical solution for four-wheel-drive (4WD), in order to operate on one axle with EM (typically rear axle) and on the other axle with ICE (typically front axle)
- *Double shaft system:* the connection is at transmission level.
- *Single shaft system:* the connection is at motor level; this is the most used configuration.

The first two solutions have two transmissions (causing more weight and more costs), the advantage is that can be chosen the best operating points from EM and ICE working separately; instead in the third solution (single shaft), there is less flexibility.

The Single shaft system can further be classified into (*figure 6*):

- *Coaxial solution*: this solution allows to integrate the ICE side clutch under the rotor bore in order to limit the powertrain length. This is guaranteed with the usage of electric motor in disc shape ($\lambda < 0.2$) with high pair pole numbers to increase the torque. The two actuators rotate at the same engine speed.
- *Non-coaxial solution*: in this solution, the two actuators are not on the same axle, but t is used metallic chain applied to the engine flywheel with the clutch integrated. This solution allows the usage of e-machine with $\lambda > 1$ usually with a low pair pole number (2 or 3 for passenger car applications).

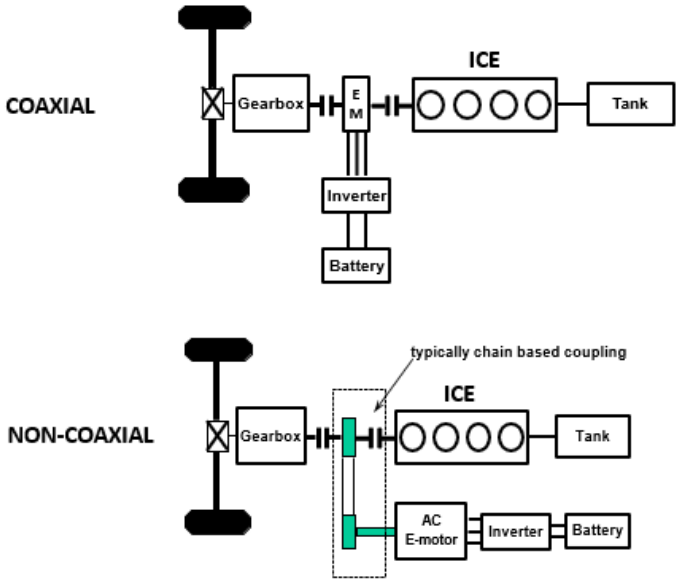


*Figure 6. Single shaft: Coaxial and non-coaxial solution*

- *P-method*

This method is applied to the simple parallel-HEV in which there is a clear distinction between engine front and engine rear, and the mechanical transmission and differential unit are not integrated in one housing, but they are decoupled (*figure 7*):

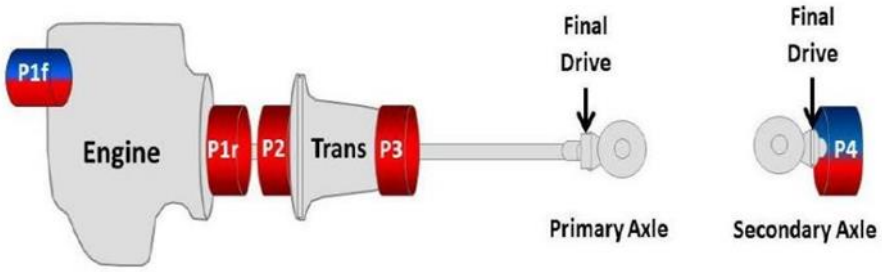


*Figure 7. P-method for simple parallel-HEV*

This P-method is classified based on the position of e-machine and it is typically applied to the RWD (rear-wheel-drive) vehicles with longitudinal engine layout:

- *P1f*: the e-machine is replacing the conventional alternator. There is a reduction-multiplying ratio (pulleys and belt coupling) between ICE and e-machine (typically 1:3 or 1:4).
- *P1r*: the e-machine is replacing the flywheel with a direct 1:1 speed ratio.
- *P2*: it is an extension of *P1r* thanks to the second clutch on the ICE side (in this case the ICE needs anyway a passive flywheel even if smaller than the one of an Internal combustion engine vehicle (ICEV) application.
- *P3*: the e-machine is connected to the transmission shaft between mechanical transmission and the rear differential unit (this is a typical solution for Commercial vehicles).
- *P4*: the e-machine.
- on the “other axle” (other in respect of the ICE propelled one).

The P-method can be also applied to the FWD (Front-wheel drive) vehicle with transversal engine layout:

- *P1f*: e-machine as auxiliary drive engine side.
- *P1r*: e-machine as the engine transmission side.
- *P3*: not possible solution in transversal layout transmission since the differential unit is fully integrated in the transmission housing.
- *P4*: e-machine on the rear axle.

## 1.3. Energy management strategy for HEV

The energy management strategy (EMS) is fundamental for hybrid electric vehicles (HEVs) since it plays a decisive role on the performance of the vehicle (fuel consumption FC and the state of charge SOC). The goal in HEV study is to minimize the fuel consumption of the vehicle maintaining the state of charge within a certain range [0,55−0,65] during the cycle, the initial SOC ($SOC_{in}$) has to be equal to the SOC at the end of driving cycle ($SOC_{end}$).

However, design a highly efficient EMS is still a challenging task due to the complex structure of HEVs and the uncertain driving mission.

The existing EMS methods can be generally classified into the following three categories (*figure 8*):

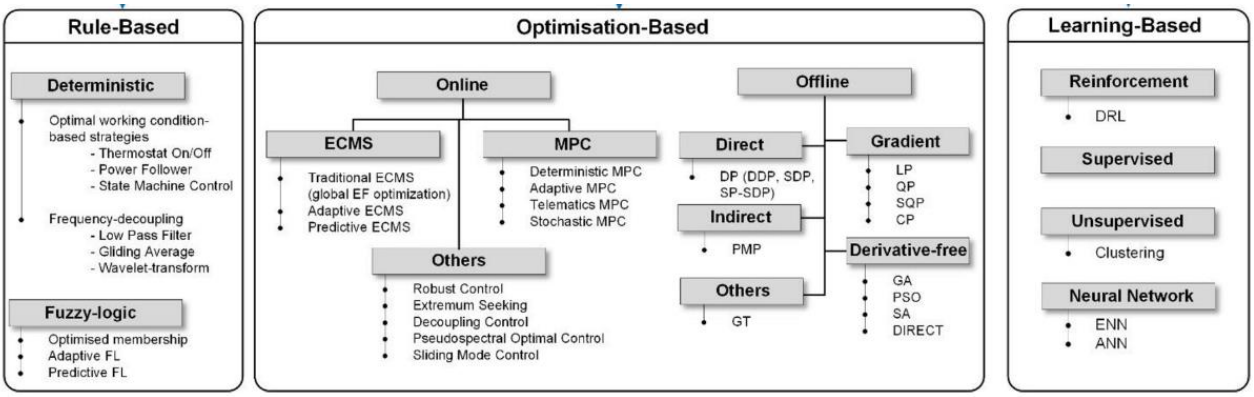


*Figure 8. Classification of energy management strategy (EMS)*

- **Rule-based EMS**: these strategies can be easily implemented, but the flexibility is critically limited by working conditions and, consequently, are not adaptive to different driving cycles.
- **Optimization-based EMS**: these strategies can find the optimal power-split solution respect to a specific driving cycle, not for all the driving mission possibilities. These methods suffer from the “curse of dimensionality” problem, which prevents their wide adoption in real-time applications. In this strategies, global optimum solutions can be obtained by performing optimization over a predefined driving cycle. So, it is not possible to perform real-time energy management. In any case, the results of these optimum solutions can be used to benchmark other control strategies and as a basis to define rules for online implementation [4].
- **Learning-based EMS**: these strategies can manage complex environmental and can be easily implemented in the software of ECU. Reinforcement learning (RL) can manage all the difficult computational, managing complex variable without computational cost [5] (*figure 9*).

| Dynamic Programming | Equivalent Consumption Minimization Strategy | Reinforcement Learning |
|---|---|---|
| **Driving Mission** known in advance | **SOLVED BUT** not for equivalent factors tuning | **SOLVED** |
| **Backward** problem | **Forward** problem | **Forward** problem |
| **Global** optimization | **Instantaneous** optimization | **Global** Optimization |
| Not suitable for **real time / online** implementation | **SOLVED BUT** not optimal if implemented on current ECUs | **SOLVED** |
| Limited to **discrete** state / control actions | **SOLVED BUT** continuous space not compatible with current ECUs | **SOLVED with advanced algorithms** |

*Figure 9. Comparison between EMS*

The Optimisation-based control strategy can be sub-divided into two categories:

- ***Offline strategies***: they require knowledge of the entire driving cycle. they can be seen as a good analysis, design and assessment tool for other types of strategies; due to their computational complexity, they are not directly implementable for real-time operation:
    1. *linear programming (LP)*: LP solves the problem of fuel consumption optimization by approximating a convex nonlinear optimization problem with a linear programming method. Its problem is that the approximate formulation restricts its application to simple series HEV architectures.
    2. *dynamic programming (DP)*: uses a numerical or analytical model to compute the optimal control strategy to achieve the best fuel consumption. It is able to deal with nonlinearity to find the global optimal solution.
    3. *metaheuristic search methods*: solve optimization problems using stochastic search techniques that reproduce natural processes (e.g., genetic algorithms, particle swarm optimization and simulated annealing).

They are effective to solve complex optimization problems with nonlinear, multimodal, and non-convex objective functions. All these offline optimization strategies cannot be used in real-time applications but can provide a benchmark for design and comparison.

- ***Online strategies***: Online strategies are used when real-time analysis is required. In these strategies, the global criterion of global optimization techniques has to be reduced to an instantaneous optimization, introducing a cost function that only depends on the present state of the system parameters.
    1. *ECMS*: an equivalent fuel factor is calculated, expressed as the actual fuel consumption that is required to recharge the batteries and to recover the energy of the regenerative braking. The total equivalent fuel consumption is the sum of the real fuel consumption of the internal combustion engine (ICE) and the equivalent fuel consumption of the electric motor. An instantaneous cost function can be calculated and minimized without the necessity for future predictions. The disadvantage of this strategy is that it does not guarantee charge sustainability ($SOC_{in} = SOC_{end}$).
    2. *model predictive control (MPC):* this control strategy in first step calculates the optimal inputs over a prediction horizon to minimize the objective function subject to the constraints; then it implements the first element of the derived optimal inputs to the physical plant; finally, it moves the entire prediction horizon forward and repeats the first step.

## 1.4. Work target

This work is focused on the study of DDQN algorithm and on the sensitivity of the immediate reward.

The double deep-Q-network (DDQN) has been developed to overcome the overestimation bias, a defect present in the DRL strategies such as Deep-Q-network (DQN). The DDQN is considered as a groundbreaking work in the field of Deep Reinforcement Learning.

The DDQL-based strategy shows more promising performance than DQL on convergence rate and policy searching ability, since that DQL algorithm is generally vastly overoptimistic about the iterative value of greedy policy.

In addition to this, this work is focused on the reward function comparing different solutions that highly affects the performance of DRL. The optimization objective is the vehicle fuel economy, the reward represents a crucial function since it is designed in order to simultaneously control the fuel consumption (FC) of the internal combustion engine (ICE) and the state of charge of the battery (SOC)

The sensitivity analysis on the five reward functions analysed shows how calibrating correctly this crucial function of the algorithm allows to improve the effectiveness of the reward having a better performance of the DRL algorithm and maximize the goal of the energy management strategy:

$$R_t = f(FC(t), SOC(t))$$

The DDQN and the different reward functions analysed are tested on four different driving cycles (urban, extra-urban, highway and mixed driving cycles) that cover most possible driving scenario of a passenger car.

# 2. Vehicle model

For this case study, the vehicle chosen is a parallel P2-HEV full performance passenger car. The layout refers to the driveline of a hybridized front-wheels driven passenger car equipped with a conventional downsized ICE and a 6-gears transmission (*Figure 10*).

## 2.1 Vehicle architecture

More specifically, it is a simple-thermal electric parallel hybrid (P2-HEV) with two torque actuators (ICE and MG) in which the hybridization is obtained through mechanical link (Torque coupling device TCD and clutches C1 and C2):

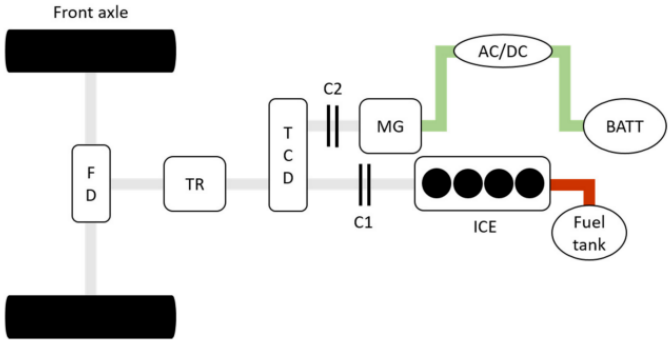


*Figure 10. Powertrain configuration of parallel P2-HEV (case study)*

This architecture (shown in *figure 10*) allows the possibility to use four macro-operating modes [6]:

- **Pure Thermal mode (PT):** engaging the clutch C1 while disengaging the clutch C2 so that the ICE can be used as a stand-alone component.
- **Pure Electric mode (PE):** engaging clutch C2 while disengaging clutch C1 so that the motor-generator (MG) and the battery (BATT) can be used as a stand-alone component.
- **Power-split mode (PS):** engaging both clutches C1 and C2 in order to demanding power to both the propellers. In this work are considered three different power-split ($\alpha$=0.25/0.5/0.75) in which $\alpha = \frac{P_{EM}}{P_{req}}$.
- **Battery charging mode (BC):** engaging both clutches C1 and C2 using the MG as generator with the capability of transferring a given share of power to the battery even during traction.

The vehicle data are listed into *table 1:*

| Vehicle | | |
|---|---|---|
| ***Paramaters*** | ***value*** | ***unit*** |
| *Kerb weight* | *750* | *[kg]* |
| *Vehicle mass (with powertrain components)* | *1200* | *[kg]* |
| *Tyre diameter* | *0,6014* | *[m]* |
| *wheel inertia* | *1,05* | *[kg * m^2]* |
| *Rolling resistance* | *0,0879* | *[N/kg]* |
| *Aerodinamic drag resistance coefficient (Cx)* | *0,27* | *[/]* |
| *Frontal area* | *2,19* | *[m^2]* |
| *Front/Rear braking distribution* | *0,75* | *[/]* |

*Table 1. vehicle data*

The efficiencies have been considered for each powertrain component (ICE and MG) including final drive (FD), multi-gears transmission (TR), torque-coupling device (TCD) and inverter (AC-DC) considering fixed efficiency to be paid both during traction and braking phases.

Instead, the transmission ratio (TR) has been modelled according to different efficiency for each gear.

## 2.2 Vehicle model (kinematic backward approach)

A kinematic backward approach (*figure 11*) has been used to compute the power requested to the different stages of the powertrain (from the traction power requested by driving cycle to the requested by power components):

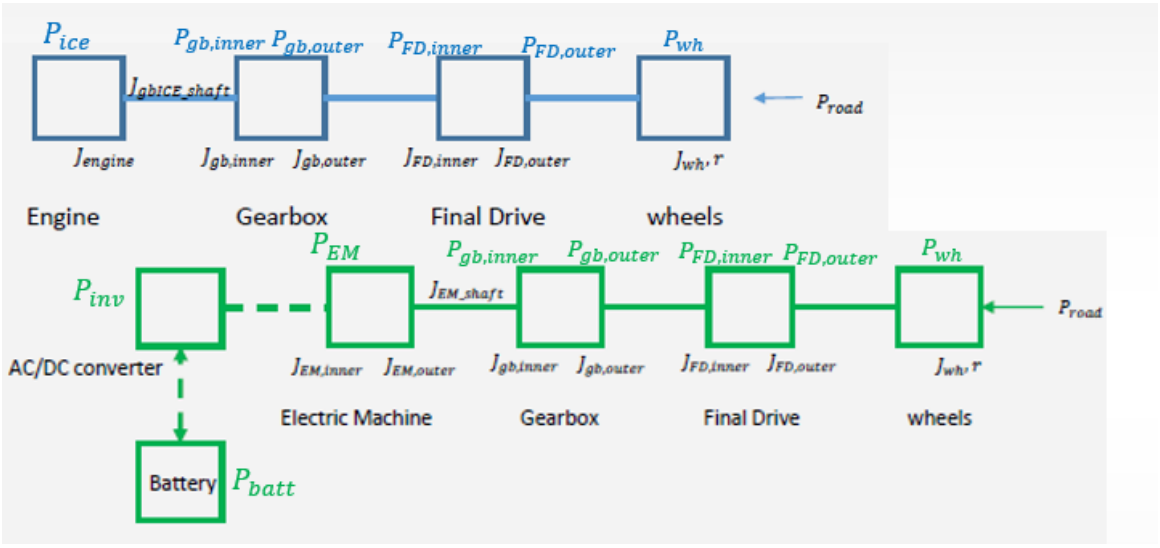


*Figure 11. OD-kinematic model for ICE (up) and EM (down)*

The driving cycle is known as input, it produces a torque at wheel, the torque is managed by the powertrain model and produces a requested torque/power, this torque is sent to the controller that decide the power split (*figure 12*):

- Torque to fuel maps $T_{ICE}$ provided FC that enters in the objective function.
- Torque to EM $T_{EM}$ enters in the battery model and produced the SOC (state variable).

With this approach, everything is managed with maps-based, except the battery model (time history of SOC).

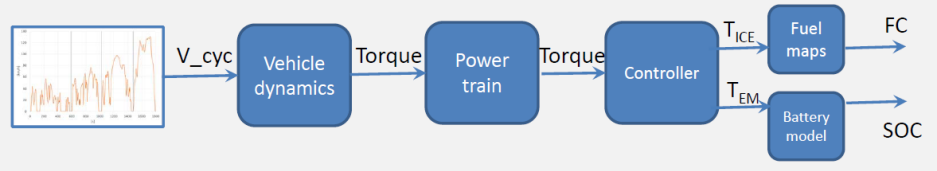


*Figure 12. Kinematic vehicle simulator (quasi-static simulation)*

The kinematic backward approach is analysed in following:

Computing the required traction power $P_{trac}(t)$ given by the speed profile and by the resistive power $P_{res}(t)$ acting on the longitudinal dynamics of the vehicle and sum of the aerodynamic force and rolling resistance force (the slope is assumed to be 0 and so the slope resistive force is null):

$$P_{trac}(t) = m * v(t) * \frac{dv(t)}{dt} + P_{res}(t)$$

This allows to compute the wheel power and speed using vehicle's apparent mass model through the ICE and MG components and evaluating the power requested by the power sources ($P_{req}$).

$$P_{req} = P_{gb_{inner}} + \mathrm{P}_{inertia,\,ICE} + \mathrm{P}_{inertia,\,EM} + \mathrm{P}_{GB,\,iceshaft}$$

Splitting the power demand between ***ICE*** ($P_{ICE}$) and ***EM*** ($P_{EM}$) using the coefficient $\alpha$ and the different operating modes allowed in a parallel P2-HEV configuration (*table 2*):

$$\begin{cases} P_{ICE} = (1-\alpha) * P_{req} \\ P_{EM} = \alpha * P_{req} \end{cases}$$

where: $\alpha = \frac{P_{EM}}{P_{req}}$

| $\alpha$ | mode |
|---|---|
| $\alpha = 1$ | Pure electric |
| $\alpha = 0$ | Pure thermal |
| $\alpha = 0{,}25 - 0.5 - 0.75$ | Power split |
| $\alpha = 0.5$ | Battery charging |

*Table 2. alpha configuration depending on the driving mode*

Finally, it is possible to evaluate the two final outputs of the model: fuel consumption (FC) by the ICE and SOC (state of charge of the battery).

### 2.2.1 Internal combustion engine (ICE) model

In order to compute fuel consumption and the motor efficiency of the ICE is experimentally derived two-dimensional look-up:

$$\dot{m}_{fc} = \dot{m}_{fc}(P_{ICE},\ \omega_{ICE})$$

The experimental data allows to know the speed-power map of the ICE, the engine chosen for the hybridization is a gasoline engine, the characteristics are shown in the *table 3*:

| ICE | | |
|---|---|---|
| ***Paramaters*** | ***value*** | ***unit*** |
| *ICE displacement* | *1* | *[KL]* |
| *ICE Inertia coefficient* | *0,14* | *[kg * m^2]* |
| *minimum engine speed* | *1000* | *[rpm]* |
| *maximum engine speed* | *6250* | *[rpm]* |
| *Maximum power* | *88* | *[kW]* |
| *Maximum torque* | *180* | *[Nm]* |
| *fuel density* | *0,78* | *[0,78]* |
| *Lower heating value (LHV)* | *43,4* | *[MJ/kg]* |

*Table 3. ICE parameters*

### 2.2.2 Electric motor (EM) and inverter models

For the electric motor (EM), the efficiency is experimentally derived two-dimensional look-up:

$$\eta_{EM} = \eta_{EM}(w_{EM},\ T_{EM})$$

Instead for the EM, the experimental data allows to know the speed-torque map in both side (motor and generator), the characteristics are shown in the *table 4*:

| EM | | |
|---|---|---|
| ***Paramaters*** | ***value*** | ***unit*** |
| *Maximum power* | *70* | *[kW]* |
| *Maximum torque* | *154* | *[Nm]* |
| *Rotational speed range* | *0-13500* | *[rpm]* |
| *EM inertia coefficient* | 0,015 | *[kg * m^2]* |
| **Inverter** | | |
| ***Paramaters*** | ***value*** | ***unit*** |
| *AC/DC converter efficiency* | *0.95* | *[/]* |

*Table 4. EM and inverter parameters*

### 2.2.3 Li-ion Battery model

The battery used in the vehicle are lithium-ion based, cylindrical type.

The battery performances have been accounted for by a zero-order Rint model (*figure 13*), static model in which the effect of environmental temperature, battery temperature and aging process have been ignored:

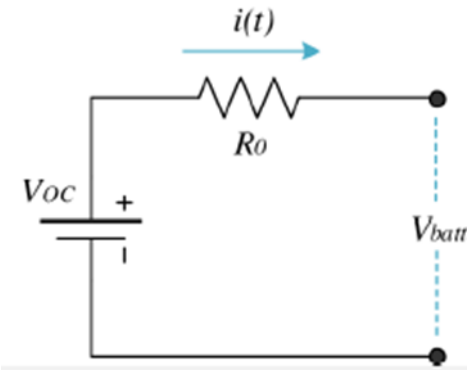


*Figure 13. Internal resistance model of the battery (Rint model)*

In this model, the internal resistance $R_0$ and the open circuit voltage $V_{OC}$ are only functions of the state of charge (*SOC*).

The maximum power achieved by the battery is linked to both electric and chemical phenomena.

The internal parameters of the battery have to be evaluated at each battery SOC level; the discharging/charging current has to be evaluated according to the following formulation:

- In traction:

$$P^{tr}_{batt,\,max,\,el}(SOC) = \frac{V^2_{OC}(SOC)}{4 * R_{eq}(SOC)}$$

$$P^{tr}_{batt,\,max,\,chem}(SOC) = V_{OC}(SOC) * I_{max,dis} - R_{eq}(SOC) * I^2_{max,\,dis}$$

$$P_{batt,\,max}(SOC) = min\left(P^{tr}_{batt,\,max,\,el}(SOC), P^{tr}_{batt,\,max,\,chem}(SOC)\right)$$

- In braking:

$$P^{br}_{batt,\,max}(SOC) = -\left(V_{OC}(SOC) * I_{max,ch} + R_{eq}(SOC) * I^2_{max,ch}\right)$$

Where:

- $I_{max,dis} = C_{batt} * C_{max,\,dis}$
- $I_{max,ch} = C_{batt} * C_{max,\,ch}$
- $C_{batt} = \frac{E_{batt}}{V_{batt,\,nom}}$

After evaluated the battery required power in traction and braking, the battery SOC variation and the current of the battery can be computed as follow:

$$SOC_t = SOC_{t-1} - \int \frac{I_{batt}}{C_{batt}} * dt \rightarrow I_{batt} = \frac{V_{OC}(SOC) - \sqrt{V_{OC}^2(SOC) - 4 * R_{eq}(SOC) * P_{batt}}}{2 * R_{eq}(SOC)}$$

The architecture of the Li-Ion battery is shown in the *figure 14*:

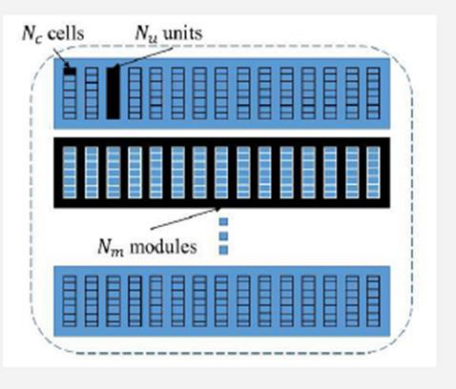


*Figure 14. Typical architecture of high voltage Li-Ion battery*

Knowing the configuration of the battery (number of units in parallel $N_{pu}$ and number of modules in series $N_{sm}$, the curves of the battery equivalent internal resistance and open circuit voltage can be evaluated:

$$\begin{cases} R_{eq}(SOC) = R_{eq,ref}(SOC) * \dfrac{N_{sm} * N_c}{N_{pu}} \\ V_{OC}(SOC) = V_{OC,ref}(SOC) * N_{sm} * N_c \end{cases}$$

Where:

- $N_{pu} = \frac{C_{batt}}{C_u}$
- $N_{sm} = \frac{V_{batt,nom}}{V_m}$

The complete configuration of the battery is reported in the *table 5*:

| **Battery** | | |
|---|---|---|
| ***Paramaters*** | ***value*** | ***unit*** |
| *Peak power* | *74* | *[kW]* |
| *Specific energy (Ebatt)* | *14,1* | *[kWh]* |
| Battery capacity | 20,29 | [Ah] |
| *Battery nominal voltage (Vbatt,nom)* | *300* | *[V]* |
| *Cell nominal voltage* | *3,6* | *[V]* |
| *Cell capacity* | *2,9* | *[Ah]* |
| *Number of cells in a unit (Nc)* | *6* | *[/]* |
| *Number of unit in parallel (Npu)* | *7* | *[/]* |
| *Number of modules in series (Nsm)* | *14* | *[/]* |

*Table 5. battery parameters*

### 2.2.4 Gearbox, final drive and torque coupling device (TCD) models

In the gearbox model has been considered experimental data: considering six gears ratio, the efficiency of the transmission in the corresponding gear ratio selected, the gearbox momentum of inertia and the gearbox mass.

The gearbox parameters chosen are listed in the *table 6*:

| Gearbox | | | |
|---|---|---|---|
| ***Paramaters*** | ***value*** | | ***unit*** |
| | ***Gear number*** | ***Gear efficiency*** | |
| *first gear* | *4,17* | *0,945* | *[/]* |
| *second gear* | *2,13* | *0,956* | *[/]* |
| *third gear* | *1,32* | *0,969* | *[/]* |
| *fourth gear* | *0,95* | *0,965* | *[/]* |
| *fifth gear* | *0,75* | *0,954* | *[/]* |
| *sixth gear* | *0,62* | *0,953* | *[/]* |
| | | | |
| *Gearbox inertia coefficient* | | *0,15* | *[kg * m^2]* |
| *Outer (wheel side) gearbox inertia coefficient* | | *0,03* | *[kg * m^2]* |
| *Inner (power train side) gearbox inertia coefficient* | | *0,03* | *[kg * m^2]* |
| *Gearbox mass* | | *50* | *[kg]* |

*Table 6. gearbox parameters*

Instead, the final drive and torque coupling device (TCD) parameters are listed in the *table 7*:

| Final drive and torque coupling device | | |
|---|---|---|
| ***Paramaters*** | ***value*** | ***unit*** |
| *Outer (wheel side) final drive inertia coefficient* | *0,004* | *[kg * m^2]* |
| *Inner (powetrain side) final drive inertia coefficient* | *0,002* | *[kg * m^2]* |
| *Final drive efficiency* | *0,98* | *[/]* |
| *Torque coupling device efficiency* | *1* | *[/]* |

*Table 7. final drive and torque coupling device parameters*

# 3. Algorithm

The algorithm analysed in this work is a Double Deep-Q-network (DDQN). This algorithm is a Model-free Deep Reinforcement learning (DRL) algorithm. The DRL is a subcategory of Reinforcement learning. In the following section, the fundamental concepts of Reinforcement learning will be analysed.

## 3.1 Key concepts of Reinforcement learning

The purpose of reinforcement learning (*figure 15*) is to make optimal decisions through trial-and-error based on observations and analysis of system behaviour to improve system performance. The basic idea is to learn the optimal strategy for achieving the goal by maximizing the accumulated reward value obtained by the agent from the environment [14].

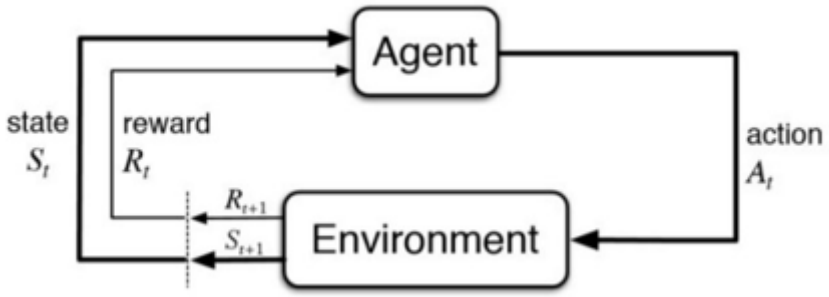


*Figure 15. Basic concept of Reinforcement Learning*

At every time step, the agent receives an observation $O_t$, that is equivalent to the state $S_t$, and a reward $R_t$. Then, the agent performs an action $a_t$ that modified the envoirnment from the state $S_t$to the state $S_{t+1}$, during this transition it is associated a new reward $R_{t+1}$ in numerical scalar value. This cycle is finished with the episode and the agent's goal is to maximize the total amount of scalar reward (*Cumulative reward*).

### 3.1.1. MDP (Markov decision process)

A learning-based method needs a formal description of the environment. In Reinforcement Learning, it is usually assumed that the environment can be described *by Markov decision process* (MDP) [8].

The MDP is a tuple $< S, h, A, P, r, y >$:

- S: it is a finite set of states.
- h: it is a function from state space to observation.
- A: it is a finite set of actions that can be chosen in the state space *S.*
- P: it is a state transition probability -> $P_a(s, s') = P_r(s_{t+1} = s' \mid s_t = s, a_t = a)$.
- r: it is a reward function that indicates the reward received using the action *a* to pass from state $s_t$ to $s_{t+1}$.
- $\gamma$: it is a *discount factor.*

A state $s_t$ can be considered Markovian state if:

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \dots, s_t]$$

All the relevant actions in the past are captured in the current state.

### 3.1.2. Return

In Reinforcement learning problems, the goal is to find *optimal policy* that maximizes the sum of all the rewards obtained during the episodes. For this reason, it is necessary to introduce the concept of *Return.*

In a finite horizon (*T*), the Return $G_t$ is defined as the sum of immediate reward $r_t$ obtained in the time instant *t:*

$$G_t = \sum_{t=0}^{T} r_t$$

But mathematically the control strategy of the HEV can be formulated as an infinite horizon dynamic optimization problem as follows:

$$R = \sum_{t=0}^{\infty} \gamma^t \, r(t)$$

In which:

- $\gamma \in (0, 1)$ is a discount factor that assures the infinite sum of cost function convergence.
- $r_t$: it is the immediate reward incurred by action $a_t$ at time *t.*

In this expression, the rewards are discounted by this factor $\gamma$, decreasing this factor the incidence of future reward decreases, preferring the immediate reward.

### 3.1.3. Optimal policy

The basic idea is to learn the optimal strategy for achieving the goal by maximizing the accumulated reward value obtained by the agent from the environment.

The control policy π is the distribution over the control actions *a*, given the current state *s*. The optimal value function $V_\pi(s)$ is exhibited as the finite expected discounted sum of the rewards:

$$V_\pi(s) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \mid \mathrm{s}_t = s\right)$$

Where:

- $E_\pi$: it represents the expectation return, following the policy $\pi$

It is possible to also define an *Action-value function (Q-function):*

$$Q_\pi(s,a) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \mid \mathrm{s}_t = s, a_t = a\right)$$

The agent's goal is to find *optimal policy* that maximize the *Return*. The best action is the one that optimize the Q-value at evert instant of time:

$$\pi^*(s) = argmax\, Q_{\pi^*}(s,a) \quad \forall s \in S$$

### 3.1.4. Bellmann equation

The Bellmann equation is used to express the generic policy $\pi$, summing the expected reward with the discounted factor ($\gamma$) multiplied by the Q-values of the following state ($s_{t+1}$) associated to the action $a'$:

$$Q_\pi(s,a) = E_\pi\big(r_t + \gamma Q_\pi\big(s_{t+1,}a'\big) \mid \mathrm{s}_t = s, a_t = a\big)$$

The Bellman's equation can be also expressed to define the optimal value to represent the maximum accumulative reward which can be obtained by taking the action $a_t$ in the state $s_t$:

$$Q_{\pi^*}(s,a) = E_{\pi^*}\left(r_t + \gamma \max_{a'} Q_{\pi^*}\big(s_{t+1,}a'\big) \mid \mathrm{s}_t = s, a_t = a\right)$$

The Bellmann's optimality equation defined the Q-values in the optimal policy $\pi^*$ as the immediate reward for the state $s_t$ and action $a_t$ added the maximum possible value among all the actions taken in the next state multiplied by the discounted factor $\gamma$.

### 3.1.5. Exploitation/Exploration strategy

One of the key factors in a Reinforcement Learning problem is the determination of the policy $\pi$ in which the algorithm selects the actions to be performed.

Usually in Reinforcement Learning, it is used the $\varepsilon - greedy\ policy$ that consists of choose the best move with $1 - \varepsilon$ probability and to choose a random move with probability $\varepsilon$:

$$\pi(s_t) = \begin{cases} arg\max_{a' \in A} Q(s_t, a')\ \ with\ probability\ 1 - \varepsilon \\ random\ action\ a \in A\ with\ probability\ \varepsilon \end{cases}$$

Using this policy, it is fundamental to tune the parameter $\varepsilon$ in order to have the correct relationship between exploration and exploitation guaranteeing an exploration in the initial stage of training and then having almost complete exploitation. A minimum exploration it is needed, in order that the algorithm does not remain trapped in the local maximum.

## 3.2 Model-free Deep Reinforcement learning (DRL)

The Deep-reinforcement learning (DRL) is a sub-category of Reinforcement learning (*figure 16*), combining the knowledge of reinforcement learning and neural network. This category is evolving quickly with the development of machine learning (ML) [9]:

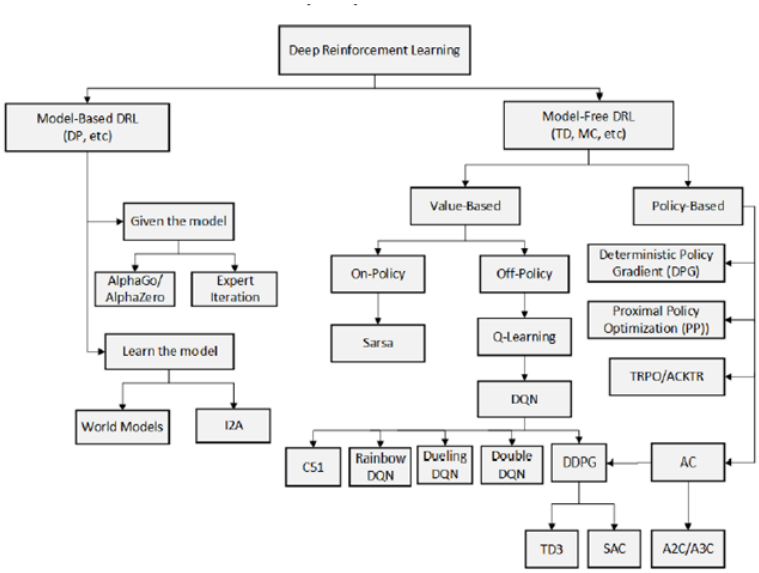


*Figure 16. Deep Reinforcement learning (DRL) classification*

The DRL is implemented as an alternative solution replacing the conventional discrete value function (Q-learning algorithm). It is innovative for three reasons:

- Firstly, the numerous inputs of matrix-form system of one state variable are replaced by a continuously changing value, resolving the “curse of dimensionality” problem.
- The discretization error is eliminated.
- Thanks to the neural network great capability of nonlinear fitting and generalization, it can well correlate the complex driving conditions and the optimal energy management strategy.

Another important feature of the DQL algorithm, with respect to the RL strategies, is the experience replay: data segments which are randomly sampled from the training data will be restored in the experience pool, and the weights of neural network will be further updated after being trained by randomization of experience information.

### 3.2.1 Q-learning

Q-learning is a famous and effective RL algorithm and has been applied in HEV energy management strategy recently.

Q-learning algorithm can give the satisfactory control orders only if action value function has been well trained. The form of action value function in Q-learning is a discretized look up-table matrix whose size is decided by dimensions of state and action variables. However, in HEV energy management problems, continuous or multi-dimensional state variables are usually needed which led the iterative computation of this matrix increase sharply and result in intractable for convergence of the training process.

The goal of Q-learning is to approximate the Q-values using the Bellmann's equation upgrading the look up-table and the temporal difference method (*TD*):

$$Q(s_t, a_t) \leftarrow Q_{value}^{(old)} - TD\ error$$

$$Q(s_t, a_t) \leftarrow Q_{value}^{(old)} + \left(Q_{value}^{(new)} - Q_{value}^{(old)}\right)$$

$$Q_{value}^{(new)} \leftarrow Q(s_t, a_t) + \alpha * \left(r_t + \gamma \max_{a'} s_{t+1}, a' - Q(s_t, a_t)\right)$$

In the last equation, $\alpha \in (0,1)$ is the *learning rate* that indicates how fast the agent is learning [7].

In complex power component structure one or two state variables may not fully represent the vehicle state in the environment and thus more state variables and more accurate grids are necessary in order to get lower fuel consumption. The property of action-value function clearly becomes a limitation of Q learning in dealing with complex problems.

Q learning-based strategy can achieve energy management task in some condition.

Nevertheless, faced with high-dimensional state space or even continuous state variables, the discretized state grid leads the matrix size increase rapidly and most likely leads to long computation time and bad convergence ability, which is also called the "*Curse of Dimensionality*" [15].

### 3.2.2 Artificial neural networks

In order to overcome the limits (approximation functions) of Q-learning, the DRL uses neural network in order to operate in continuous spaces in order to approximate the Q-values for any actions given a certain state.

The neural network (*figure 17*) is organized into:

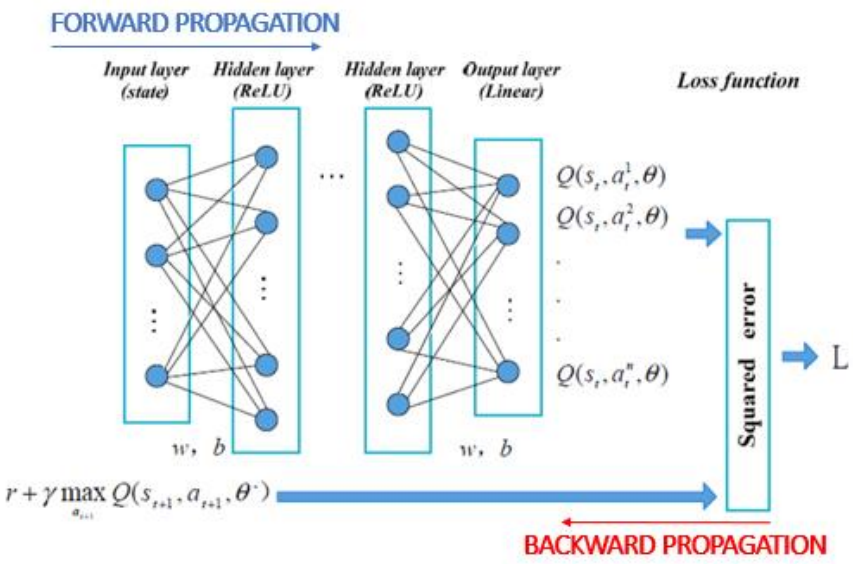


*Figure 17. Structure of the neural network*

- **Forward propagation**: the prediction process is evaluated from the input layer to the output layer, during which each neuron calculates the output value $z_j$ applying an activation function ($ReLu$) in order to correct the weights $W_i$ and bias $b$ to well predict the Q-values. The rectified linear unit is used as the activation function for hidden layers (*figure 19*):

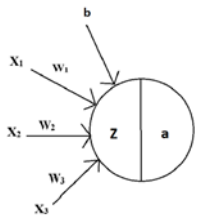


*Figure 18. Neuron of the network*

$$z_j = \sum_i W_i * X_i + b$$

$$a_j = f(z_j)$$

$$f_x = \max(0, x)$$

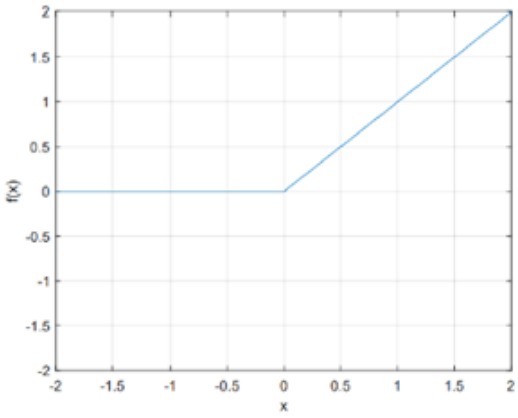


*Figure 19. ReLu activation function*

- **Backward propagation**: the optimization of neural network is based on the cost function $J$ using this squared error signal, the neural network can upgrade the weights using a gradient descent and minimize the cost function. The gradient descent (*figure 20*) changes the values of weights using the learning rate $\alpha$ that indicates the step size and controls the learning velocity: increasing it allows fast training but suboptimal weights values, otherwise decreasing it allows to obtain optimal weight values The learning rate represents one of the hyperparameters of DRL to optimize in order to obtain good performance of the algorithm extending the training time:

$$J = \left(Q_{value}^{target} - Q_{value}^{predictive}\right)^2$$
$$W_i \leftarrow W_i - \alpha \frac{\partial J}{\partial W_i}$$

Where:

- Learning rate: $\alpha \in [0,1]$

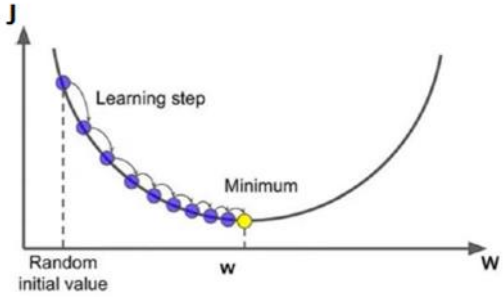


*Figure 20. Gradient descent (cost function as function of weights)*

### 3.2.3 Deep-Q-network (DQN)

The combination between the logic of reinforcement learning (RL) and the artificial neural network allows to solve complex problem such as the energy management problem of HEV since the algorithm is able to work in continuous space [10] (*figure 21*):

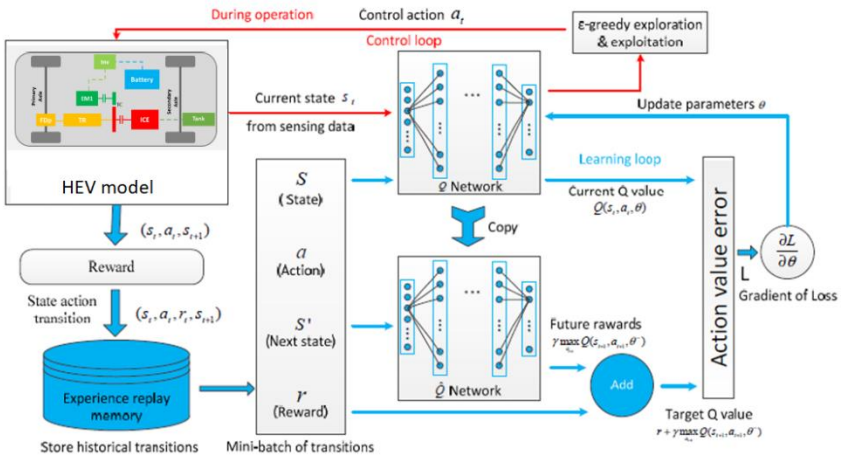


*Figure 21. DRL agent-based framework for HEV EMS*

The deep neural networks changed the original action-value function of Q-learning algorithm:

- Different from the discretization matrix, any continuous change in state variables can be reflected in a DNN-based decision system, which allows a more accurate identification of the system state without increasing computational load.
- Deep-neural network (DNN) is insensitive to the increase of state dimension numbers. These features solve the problems caused by discretization of Q-learning matrix.
- The Deep neural network has a powerful ability of function approximation, which means that the convergence speed of the action value will be greatly accelerated.

In DQN, the agent initially estimates the Q-values and explore the environment using $\varepsilon{-}greedy$ policy; progressively during the training episodes, the agent trains to predict the Q-values more and more accurately by modifying the weights through the loss function $L_t$. The mean squared error between the target Q-value and the inferred output of neural network is defined as loss function:

$$L_t = \left(E\left[r_t + \gamma \max Q(s_{t+1}, a')\right] - Q\left(s_t, a_t\right)\right)^2$$

Where:

- $Q\left(s_t, a_t\right)$: it is the output of the neural network.
- $r_t + \gamma \max Q(s_{t+1}, a')$: it is the target Q-value.

The goal is to minimize the loss function (error) in order that the network prevision get as close as possible to the desired result (expected return). Minimizing the loss function, the agent can take decisions (actions) in order to maximize the discounted return.

The main elements of Deep-Q-network are:

- **Q-network:** predict the Q-values based on certain action $a_t$ in the state $s_t$ i in the generic instant $t$. The weights are upgraded at every time in order to get the prevised value as close as possible to the desired value.
- **Q-target network**: predict the Q-target value based on the immediate reward to which the best value of Q-values among all the actions that can be chosen by the state $s_{t+1}$ is added. The weights are upgraded according to a certain frequency (*target update frequency*) in order to have stable training since, only after a certain interval, the weights trained in Q-network are copied into Q-target network.
- **Experience replay**: buffer that store some experience, each experience is a tuple ($s_t, a_t, r_t, s_{t+1}$). During the training, at every time instant, N tuples are randomly extracted in order that the network can learn weights that well generalize all the possible scenarios that the agent will have to manage. The experience replay is helping to obtain more stable training.

### 3.2.4 Double Deep-Q-network (DDQN)

The Double Deep-Q-network (DDQN) is an upgrading version of the DQN (Deep-Q-network). The algorithm's logic [..] is shown in *Figure 22 [11]:*

**Algorithm 1** DQN and DDQN

Initialize Q-network $Q_N$ and target network $O_T$ with random weights
  for each episode **do**
    for each environment step **do**
    Collect observation $O_t$ and select action $A_t$
    Execute $A_t$ and collect next observation $O_{t+1}$ and reward $R_t$
    Store tuple $(O_t,\ A_t,\ R_t, O_{t+1})$ in memory replay buffer
    Sample tuple $(O_t,\ A_t,\ R_t, O_{t+1})$ from memory buffer
    Compute loss function of the Q-network:
      $L = R_t + \gamma \max_a Q_T(O_{t+1}, a) - Q_N(O_t, A_t)$ for DQN
      $L = R_t + \gamma Q_T\left(O_{t+1}, \operatorname{argmax}_a Q(O_{t+1}, a)\right) - Q_N(O_t, A_t)$ for DDQN
    Perform gradient descent to update $Q_N$
    Every *n steps* the $Q_T$ is updated
  **end for**
**end for**

*Figure 22. DDQN algorithm*

The process of the energy management strategy based on DDQN algorithm [12] is presented in the *figure 23:*

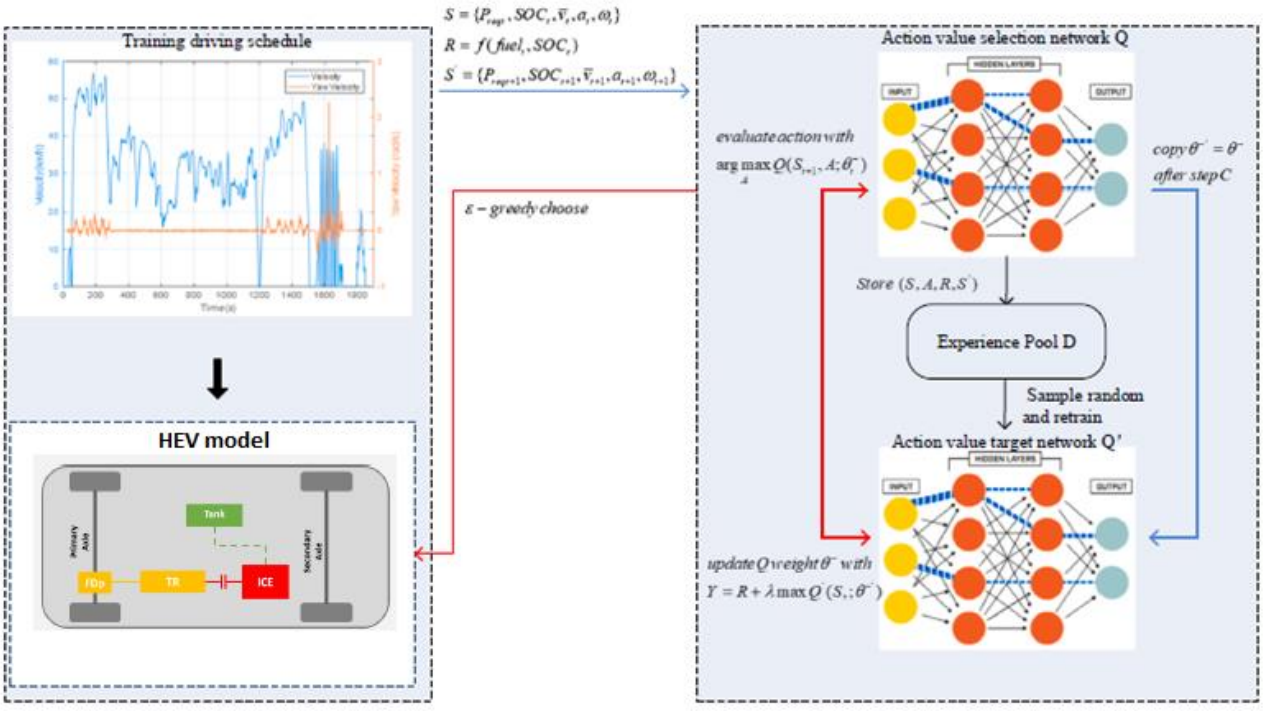


*Figure 23. Process of the energy management strategy based on DDQN algorithm*

We initialize Q-network $Q_n$and target network $Q_t$ with random weights, for each episode.

The agent is trained on multiple time steps in many episodes and performs a sequence of operations at each time step of the simulation. During the learning process, the algorithm selects the maximum Q-value action with probability 1 − ε and selects a random action with probability ε based on the observation of the state $O_t$ and a reward $R_t$ is obtained. The state action transition tuple is stored in memory replay buffer and then a mini batch of transition tuples is drawn randomly from the memory replay buffer. Computing the loss function of the Q-network, the weights in Q-network $Q_n$ are updated by using the gradient descent method, instead the target network $Q_t$ is periodically updated by copying parameters from the Q-network $Q_n$.

In DQN, the Q-value in the loss function is calculated with the reward $R_t$ added to the next state maximum Q-value so that the Q-value will become higher every time. This logic in which the Q-values are evaluated, does not allow that the neural network is upgrading if in some condition for some memory experience an action $b$ is the better than action $a$ in the state $s$.

In the DDQN, the index of the highest Q-value is evaluated in the main model, and it is used to obtain the action in the second model.

## 3.3 Software framework and configuration

The software used in this work is composed by (*figure 24*):

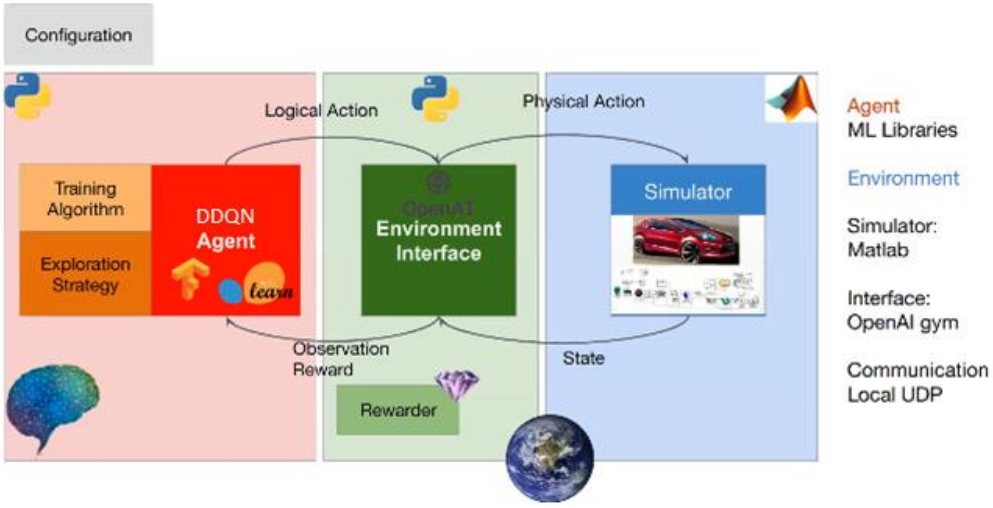


*Figura 24. Software framework configuration*

- **Simulator**: implemented in MATLAB and represents the vehicle model (physical part of the environment).
- **Environment interface and Agent**: implemented in Python.

At every physical step computed by the Simulator corresponds a logic step (step computed by the agent).

When the physical step is completed, the agent through UDP protocol send a logical state to the Environmental interface that allows the transformation between the agent and the simulator. The observation is a part of the logical state that the agent can see, the reward instead is the prize that the agent receives by the environment interface [7].

The agent has a logical interface that can be divided into:

- **Training algorithm**
- **Approximator**: artificial neural network (in this case).
- **Exploration strategy**: a component of the agent that allows to obtain optimal action-value function and find the optimal policy that guarantees the maximization of the discounted return.

After the exploration strategy, it is obtained a logical action that becomes a physical action in the simulator.

The training cycle of a DDQN's agent is divided into training and testing episodes. The test episodes are run at predefined intervals to monitor the progress of the learning in the absence of exploration components.

The important parameters for the performance of a neural network agent are:

- *learning starts*: which determines the initial part of the training, i.e., the number of steps before the network starts updating.
- *Batch*-size: the learning starts hyperparameters is in communication with a minibatch characterized by a certain size (*batch size*), so the Q network is starting to train when it can be approximately assumed that the samples within the minibatch are independent of each other.
- *replay memory buffer:* which contains a limited number of transitions $(s_t, a_t, r_t, s_{t+1}, done)$, it should be configured so that the agent forgets information that is no longer useful.
- *target update frequency*: it is the frequency in which the weights of Q-network $Q_n$ are copied within the target network $Q_t$. This parameter must be calibrated in order to guarantee a good stability of the agent. This parameter is calibrated according to the different driving cycle.

All these parameters are calibrated according to a parametric study done in a previous work [7]. A sub-optimal parametric solution using the approval driving cycle WLTP has been obtained.

The final configuration chosen is reported into *table 8*:

| | | |
|---|---|---|
| **DDQN agent** | *Funzione di attivazione* | *ReLU* |
| | *Taglia del minibatch* | *32* |
| | *Discounted factor* $\gamma$ | *0.99* |
| | *Numero di neuroni* | *64* |
| | *Numero di hidden layer* | *2* |
| | *Learning starts* | *0.002* |
| | *Target update frequency* | |
| | *Replay memory size* | *10000* |
| | *Learning rate lr* | *0.0002* |
| **Exploration strategy** | $\varepsilon start$ | *1* |
| | $\varepsilon finish$ | *0.05* |
| | $\varepsilon decay\ episode$ | *0.001* |
| **Reward** | | |
| **State** | *SOC* | |
| | *roadVeh* | |
| | *next roadVeh* | |
| **Training** | *Driving cycles* | *Clust 12Mod, Clust 11, Clust 1, Clust 19* |
| | *Episodes* | *500* |

| | | |
|---|---|---|
| Target update frequency | Clust12Mod | 1600 |
| | Clust 11 | 2000 |
| | Clust 1 | 2500 |
| | Clust 19 | 5300 |

*Table 8. Final configuration*

### 3.3.1 Control variables

The agent can be chosen two different control actions:

- Power-flow (PF)
- Gear number (GN)

The control variables are discretized in the following values:

- The power-flow number $N_{PF}$ is discretized in seven values according to the value of $\alpha$ described in the *paragraph 2.2*:
  - $\alpha = 1$: pure electric mode.
  - $\alpha = 0$: pure thermal mode.
  - $\alpha = 0.25/0.5/0.75$: power-split mode.
  - $\alpha = -0.5/-1$: battery charging mode.

- The gear number $N_{GB}$ is discretized in six values according to the gear used (from the first gear to sixth gear).

The total number of actions that the agent can choose is the combination between power-flow and gear number:

$$N_{AZ} = N_{GB} * N_{PF}$$

In addition to this, there are also Boolean variables (*feas* and *new feas*) that allows to the agent to execute only feasible subset combination of power-flow and gear number. The Boolean variables consider of the maximum speed reachable from the electric motor and internal combustion engine, consider of the maximum power absorption by the power sources and the maximum battery power allowed (according to the C-rate).

### 3.3.2 States

As it is anticipated previously, the deep-neural network allows to operate in continuous space, overcoming the limitations of the Q-learning algorithm. The state chosen in this configuration are:

- SOC
- SOC_high
- SOC_low
- Feas
- FC
- FC_norm
- battPel
- roadVel
- time
- roadPower
- roadGrade
- next_roadVel
- next_roadPower
- next_roadGrade
- action_mask

The selection of the state is one of the crucial functions of the agent since they give the information to take the decision in choosing the actions in order to maximize the *Discounted return* and improving the performance of the algorithm.

Another crucial function that affects the performance of the agent is the reward function that will be analysed in the *chapter 4*.

## 3.3 Equivalent consumption minimization strategy (ECMS)

The ECMS (Equivalent Consumption Minimization Strategy) is online optimization-based energy management strategy based on the global criterion to an instantaneous optimization problem, introducing a cost function dependent only on the system variables at the current time. In this work has been chosen as a benchmark energy management strategy to explore the fuel economy potential.

### 3.3.1 ECMS algorithm

The logic is based on instantaneous cost function that is evaluated as a sum of the fuel consumption $\dot{m}_f$ and an equivalent fuel consumption related to the SOC variation $P_{batt}(t) * \frac{s}{Q_{LHV}}$:

$$\dot{m}_{eqv} = \dot{m}_f + P_{batt}(t) * \frac{s}{Q_{LHV}}$$

The assumption behind this approach is that every variation in the state of charge will be compensated in the future by the engine running at the current operating point.

Since the electrical energy and the fuel energy are not directly comparable, an equivalence factor is needed.

The overall fuel consumption can be considered as a function of the equivalence factors and a systematic optimization can be used in order to find the equivalence factors that minimize the overall fuel consumption constrained to the SOC sustainability (final SOC equal to the initial SOC).

### 3.3.2 Optimization of equivalence factors

The tuning of the equivalence factors is needed in order to obtain the charge-sustaining solution for each driving cycle, since the local optimization is not depending on the driving mission. The choice of the equivalence factor affects two parameters:

- Charge-sustainability solution
- Effectiveness of the energy management strategy

The equivalence factors $s_{chg}$ and $s_{dis}$ must be chosen as a trade-off between the fuel consumption minimization and the charge-sustaining operating mode of the HEV model:

- If the equivalence factors are too low: it is attributed a poor cost the usage of electric energy, this means that the high-voltage battery is depleting very fast.
- If the equivalence factors are too high: it is considered an excessive cost of the usage of battery, it is not exploited the total potential of the hybridization.

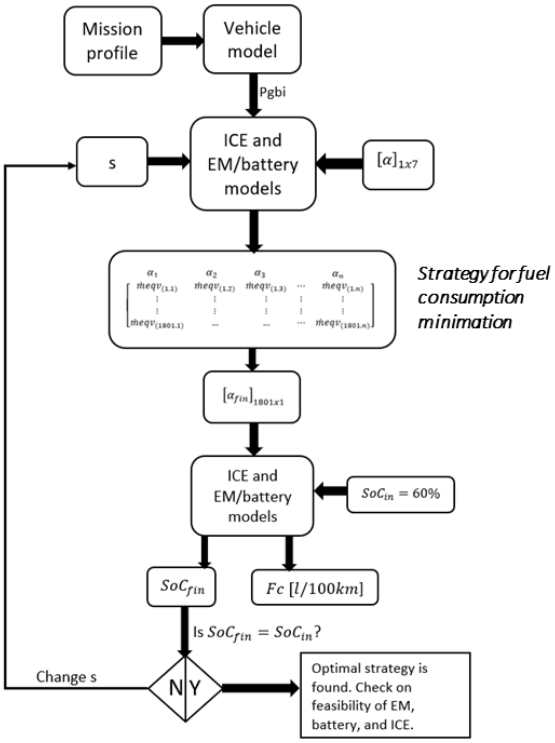


*Figure 25. ECMS flow chart*

In order to optimize the equivalence factor, has been built a matrix containing all the possible equivalent fuel consumption (*figure 25*):

- Each i-row represents the equivalent fuel consumption at the i-time of the driving cycle.
- Each j-column is the corresponding value of the power split parameter α.

By selecting, for each time instant of the cycle, the minimum equivalent fuel consumption among the n-columns, it is possible to compute the vector$\alpha_{fin}$ containing the power-split parameter associated to the minimum value of the equivalent fuel consumption. Finally, by knowing how the power is divided in each time instant using the models of internal combustion engine, electric motor and battery, the fuel consumption and the SoC associated to the value of s initially chosen can be computed.

In order to tune the optimal equivalence factor, it is used the following expression:

$$s_{k+1} = s_k + \left(SoC_k - SoC_{ref}\right) * G$$

where:

- $G = \frac{s_1}{(SoC_1 - SoC_{ref})}$
- $SoC_{ref}$= 0.6

The first value assigned to s ($s_k$) is set at 1.5 and for the following values were changed by hand based on the difference between the final and the reference state of charge.

In the following *table 9* are reported the equivalence factors after the optimization for each driving cycle:

| Equivalence factors | |
|---|---|
| **Driving cycle** | **value** |
| *Clust12Mod* | *2.307* |
| *Clust11* | *2.385* |
| *Clust 1* | *2.684* |
| *Clust 19* | *2.633* |

*Table 9. equivalence factors*

# 4. Reward function

The reward functions analysed in this work are five (*table 10*):

| REWARD | |
|---|---|
| ***Reward 1*** | *Normalized FC-oriented reward* |
| ***Reward 2*** | |
| ***Reward 3*** | |
| ***Reward 4*** | *ECMS-based Reward* |
| ***Reward 5*** | *FC-oriented reward* |

*Table 10. Reward functions analysed*

The first three are normalized FC-oriented reward based on the weighted average. The fourth reward is based on the traditional equivalent consumption minimization strategy (ECMS) logic. The fifth reward is based on alpha coefficient that determines the weight of the battery state of charge (SOC) and fuel consumption (FC).

## 4.1. Normalized FC-oriented Reward

$$if \rightarrow \begin{cases} SOC > SOC_{ref} \rightarrow \quad r_{soc} = 1 \\ SOC < SOC_{ref} \rightarrow \begin{cases} r_{soc} = 1 - 2\dfrac{\left(SOC - SOC_{ref}\right)^2}{\left(SOC_{min} - SOC_{ref}\right)^2} \\ Reward(s,a) = \dfrac{(c_{soc} * r_{soc}) + (c_{FC} * r_{FC}) + \left(c_{FC_{norm}} * r_{FC_{norm}}\right)}{c_{soc} + c_{fc} + c_{fc,norm}} \end{cases} \end{cases}$$

Where:

- $r_{fc_{norm}} = 1 - 2 * \left(\frac{FC}{\max\left(FC_t\right)}\right)$
- $r_{FC} = 1 - 2 * \left(\frac{FC}{FC_{max}}\right)$
- $FC_{max} = 0.05\ [kg]$

The three rewards are based on weighted average concept in order to generalize this function since it is crucial in order to get the simultaneous objective (minimize as soon as possible the fuel consumption and keeping the battery state of charge between the range [0,55-0,65] stabilized for HEV vehicle energy management problem.

The reward is based on:

- $c_{soc}, c_{FC}, c_{FC_{norm}}$: three offline coefficients chosen by the user.
- $r_{soc}, r_{FC}, r_{FC_{norm}}$ three reward coefficients calculated in order to penalize the reward calculated in the state $s$ with action $a$:
  - $r_{soc}$: penalyzing the reward if the SOC level is below the SOC threshold ($SOC\_ref$=0,6).
  - $r_{FC}$: penalyzing the reward using a normalized coefficient between the fuel consumption and the maximum possible fuel consumption from ICE.
  - $r_{FC_{norm}}$: penalyzing the reward using a normalized coefficient between the fuel consumption and the maximum fuel consumption in the determined time cycle.

The configuration of three reward is listed into *table 11*:

| ***Normalized FC-oriented reward*** | | | |
|---|---|---|---|
| | *c_soc* | *c_fc* | *c_fc_norm* |
| **Reward 1** | 2 | 1 | 1 |
| **Reward 2** | 1 | 1 | *0* |
| **Reward 3** | 1 | 0 | *1* |

*Table 11. Normalized FC-oriented reward (Reward 1,2,3)*

## 4.2. ECMS-based reward

The fourth reward founded in literature [13] and it is based on the traditional equivalent consumption minimization strategy (ECMS) logic in which the equivalence factors $S_{dis}$ and $S_{chg}$according to the tuning of the equivalence factors analysed in the *chapter 3.3.2.* on the different driving cycles analysed; instead $\eta_{dis}$ and $\eta_{chg}$ are supposed to be *0.95*:

$$\boldsymbol{Reward} = -m_{fuel} + 1$$

Where:

- $m_{fuel,\,batt} = \left(a\,\frac{S_{dis}}{\eta_{dis}} + (1-a)S_{chg}\eta_{chg}\right)\,\Delta t$
- $m_{fuel} = \left(m_{fuel,ICE} + m_{fuel,\,batt}\right)\Delta t$

## 4.3. FC-oriented reward

The fifth reward, based on literature [10], is based on alpha coefficient that determines the weight of the battery state of charge (SOC) and fuel consumption rate ($\frac{dfuel}{dt}$):

$$\boldsymbol{Reward} = 1 - \left(\alpha \frac{dfuel}{dt} + (1-\alpha)\,(SOC - 0.6)^2\right)$$

The optimization of the alpha coefficient is done after several testing on Clust 7 (extra-urban driving cycle analysed in previous work) using three alpha configurations (0.25/0.5/0.75).

In following are represented the results in terms of state of charge and cumulative fuel consumption (*figure 26-27-28*), cumulative reward (*figure 29-30-31*):

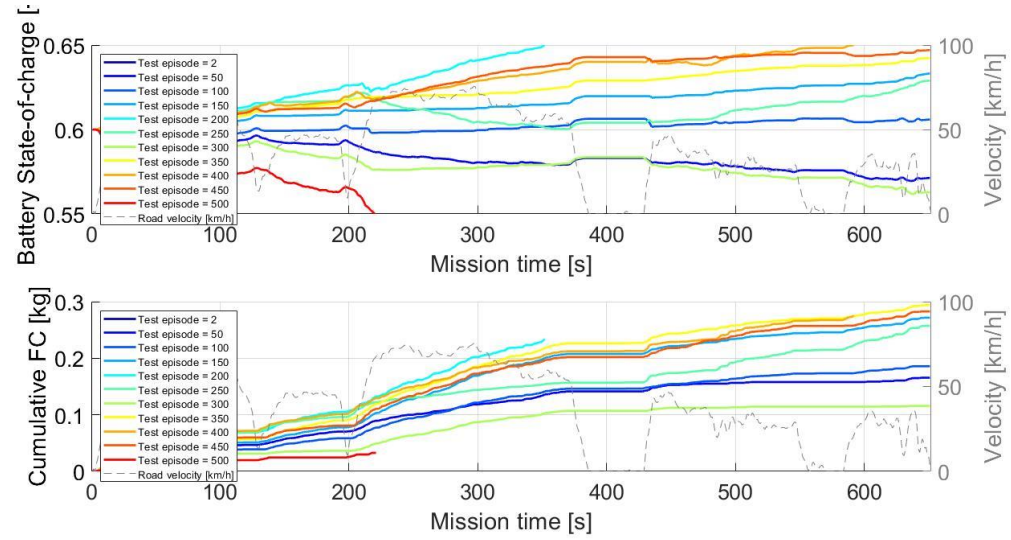


*Figure 26. SOC and cumulative FC windows (alpha=0.25)*

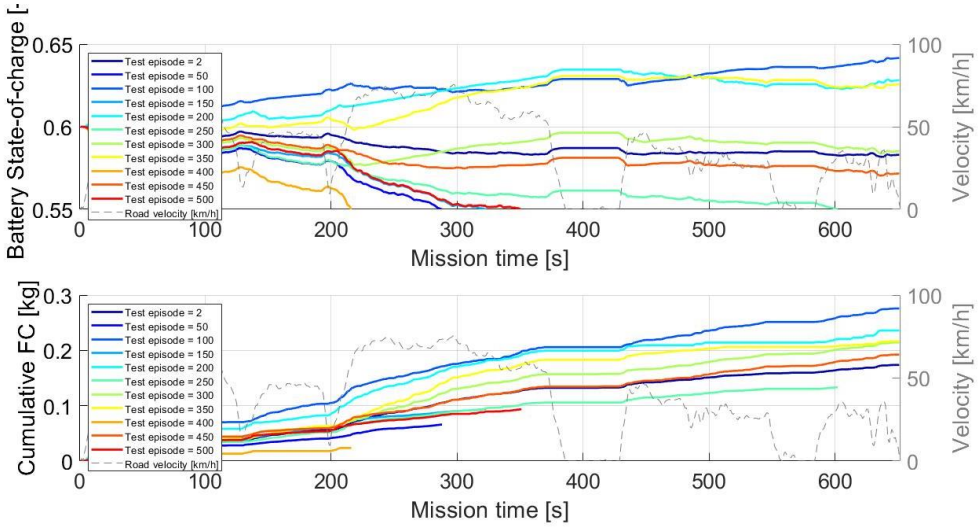


*Figure 27. SOC and cumulative FC windows (alpha=0.5)*

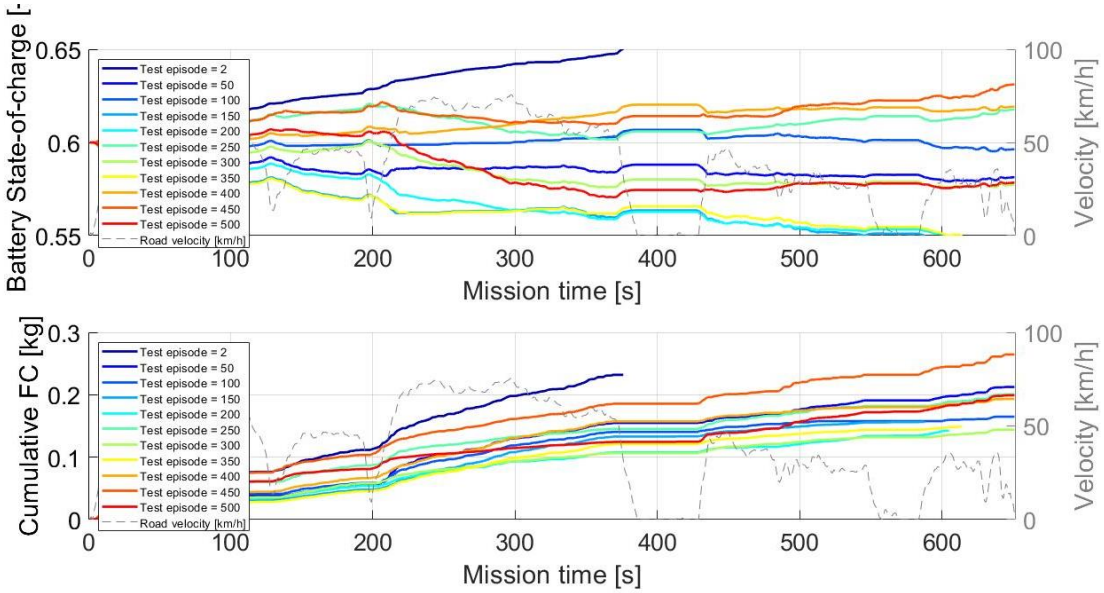


*Figure 28. SOC and cumulative FC windows (alpha=0.75)*

| Reward 5 | | |
|---|---|---|
| ***alpha*** | ***SoC [/]*** | ***cumulative FC [kg]*** |
| *0.25* | *failed* | *failed* |
| *0.5* | *failed* | *failed* |
| *0.75* | *0.578* | *0.199* |

*Table 12. SOC and cumulative FC results (Reward 5 – alpha optimization)*

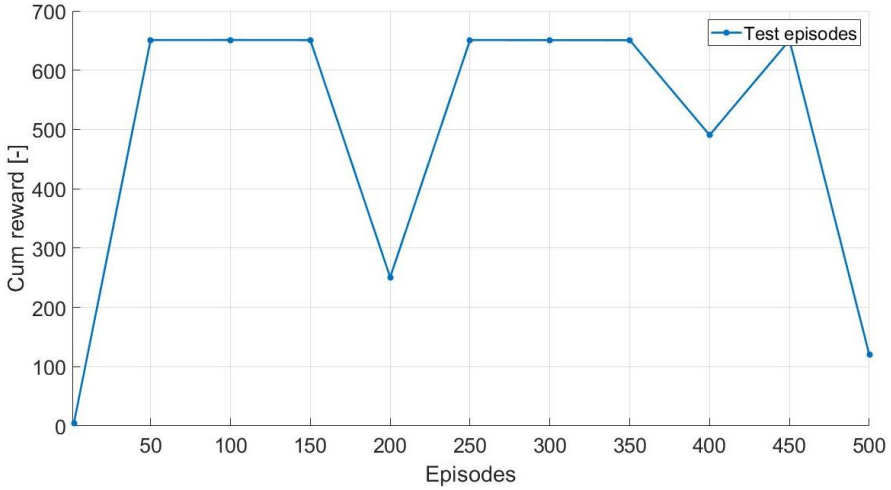


*Figure 29. Cumulative reward (alpha=0.25)*

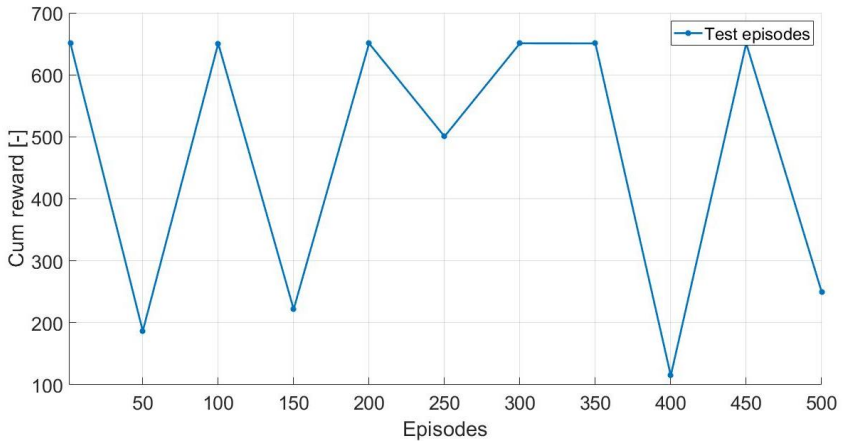


*Figure 30. Cumulative reward (alpha=0.5)*

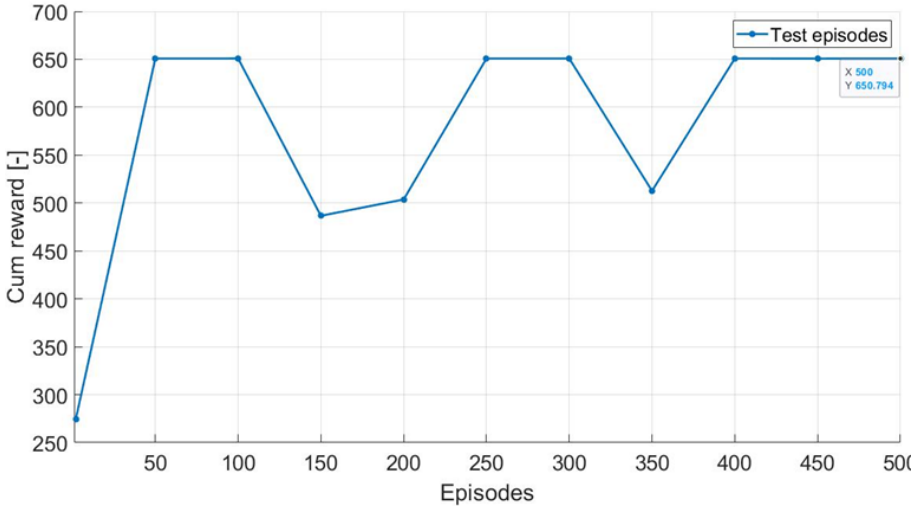


*Figure 31. Cumulative reward (alpha=0.75)*

| Reward 5 | |
|---|---|
| ***alpha*** | ***Cumulative reward [/]*** |
| *0.25* | *failed* |
| *0.5* | *failed* |
| *0.75* | *650.79* |

*Table 13. Cumulative reward results (Reward 5 – alpha optimization)*

After analysing the results, the value chosen is **alpha = 0.75.**

# 5. Driving cycles

The driving cycles analysed in this work are listed into *table 14*:

| Driving cycles | |
|---|---|
| *Urban driving cycle* | *Clust12Mod* |
| *Extra-urban driving cycle* | *Clust11* |
| *Highway driving cycle* | *Clust 1* |
| *Mixed driving cycle* | *Clust 19* |

*Table 14. Driving cycles analysed*

## 5.1. Urban driving cycles

The parameters of the urban driving cycle (Clust12Mod) are reported in the following *table 15*:

| Urban driving cycle (Clust 12Mod) | | |
|---|---|---|
| ***Parameters*** | ***value*** | ***unit*** |
| *Duration* | *400* | s |
| *Idling time* | *21* | s |
| *Average speed (with no stops)* | *22.8* | km/h |
| *Average speed (with stops)* | *19.01* | km/h |
| *max. acceleration* | *2.257* | m/s^2 |
| *min. acceleration* | *-2.67* | m/s^2 |

*Table 15. Clust 12Mod parameters*

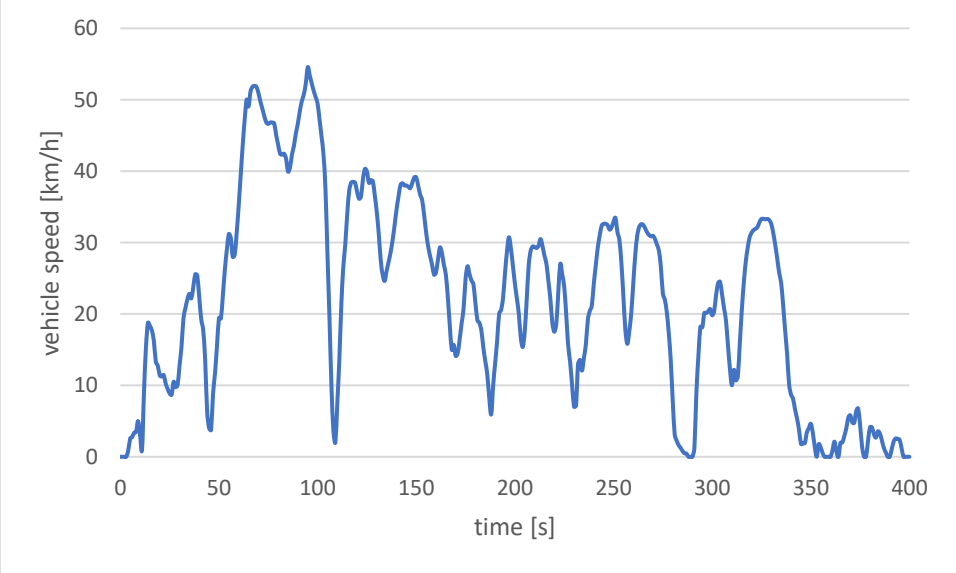


*Figure 32. Clust 12Mod*

## 5.2. Extra-Urban driving cycles

The parameters of the extra-urban driving cycle (Clust11) are reported in the following *table 16*:

| Extra-urban driving cycle (Clust 11) | | |
|---|---|---|
| ***Parameters*** | ***value*** | ***unit*** |
| *Duration* | *653* | s |
| *Idling time* | *57* | s |
| *Average speed (with no stops)* | *39.93* | km/h |
| *Average speed (with stops)* | *36.36* | km/h |
| *max. acceleration* | *2.465* | m/s^2 |
| *min. acceleration* | *-2.62* | m/s^2 |

*Table 16. Clust 11 parameters*

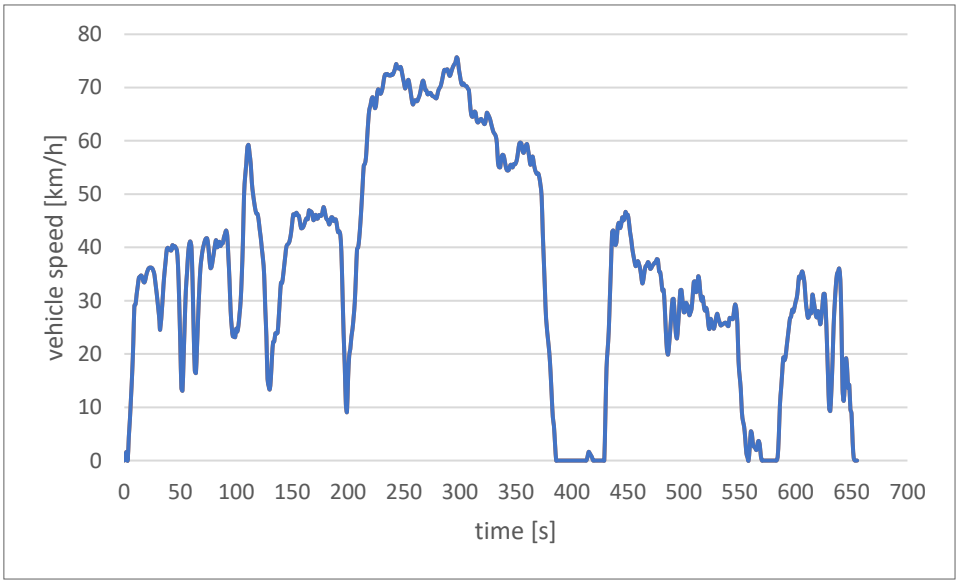


*Figure 33. Clust 11*

## 5.3. Highway driving cycles

The parameters of the highway driving cycle (Clust1) are reported in the following *table 17*:

| Highway driving cycle (Clust 1) | | |
|---|---|---|
| ***Parameters*** | ***value*** | ***unit*** |
| *Duration* | *851* | s |
| *Idling time* | *17* | s |
| *Average speed (with no stops)* | *103.71* | km/h |
| *Average speed (with stops)* | *101.71* | km/h |
| *max. acceleration* | *1.2* | m/s^2 |
| *min. acceleration* | *-1.88* | m/s^2 |

*Table 17. Clust 1 parameters*

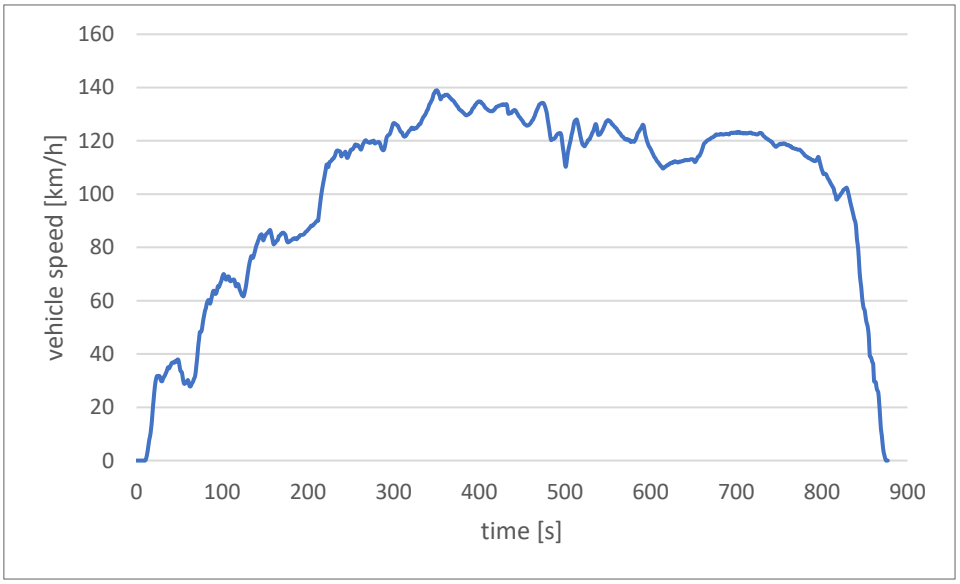


*Figure 34. Clust 1*

## 5.4. Mixed driving cycles

The parameters of the mixed driving cycle (Clust19) are reported in the following *table 18*:

| **Mixed driving cycle (Clust 19)** | | |
|---|---|---|
| ***Parameters*** | ***value*** | ***unit*** |
| *Duration* | *1935* | s |
| *Idling time* | *95* | s |
| *Average speed (with no stops)* | *66.34* | km/h |
| *Average speed (with stops)* | *63.09* | km/h |
| *max. acceleration* | *2.47* | m/s^2 |
| *min. acceleration* | *-3.51* | m/s^2 |

*Table 18. Clust 19 parameters*

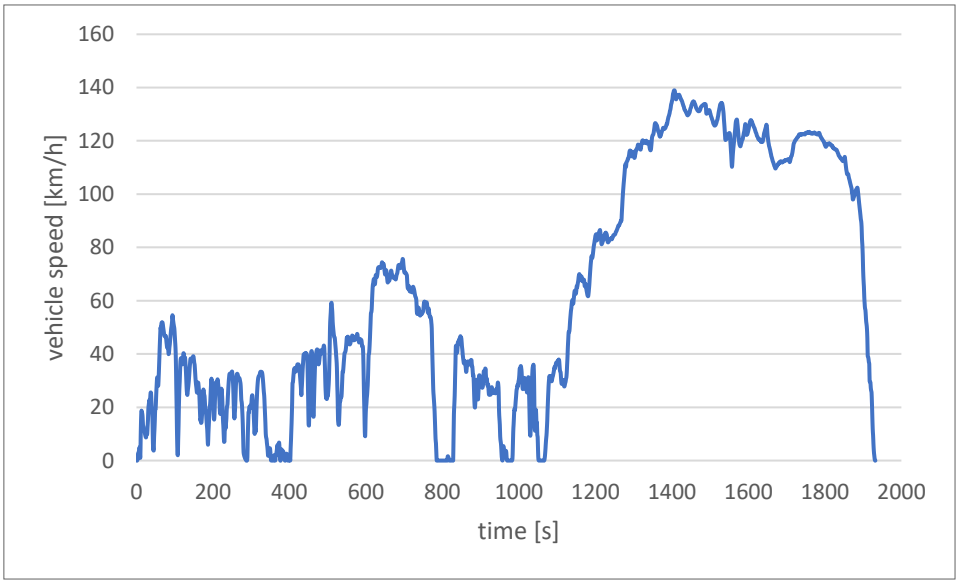


*Figura 35. Clust 19*

# 6. Results

The results of this work are represented in terms of state of charge of the battery *SOC (figure 36-40-43-46)*, cumulative fuel consumption *FC* (*figure 37-41-44-47*) and cumulative reward (*figure 38-39-42-45-48*) for each driving cycle analysed.

## 6.1. Clust 12Mod

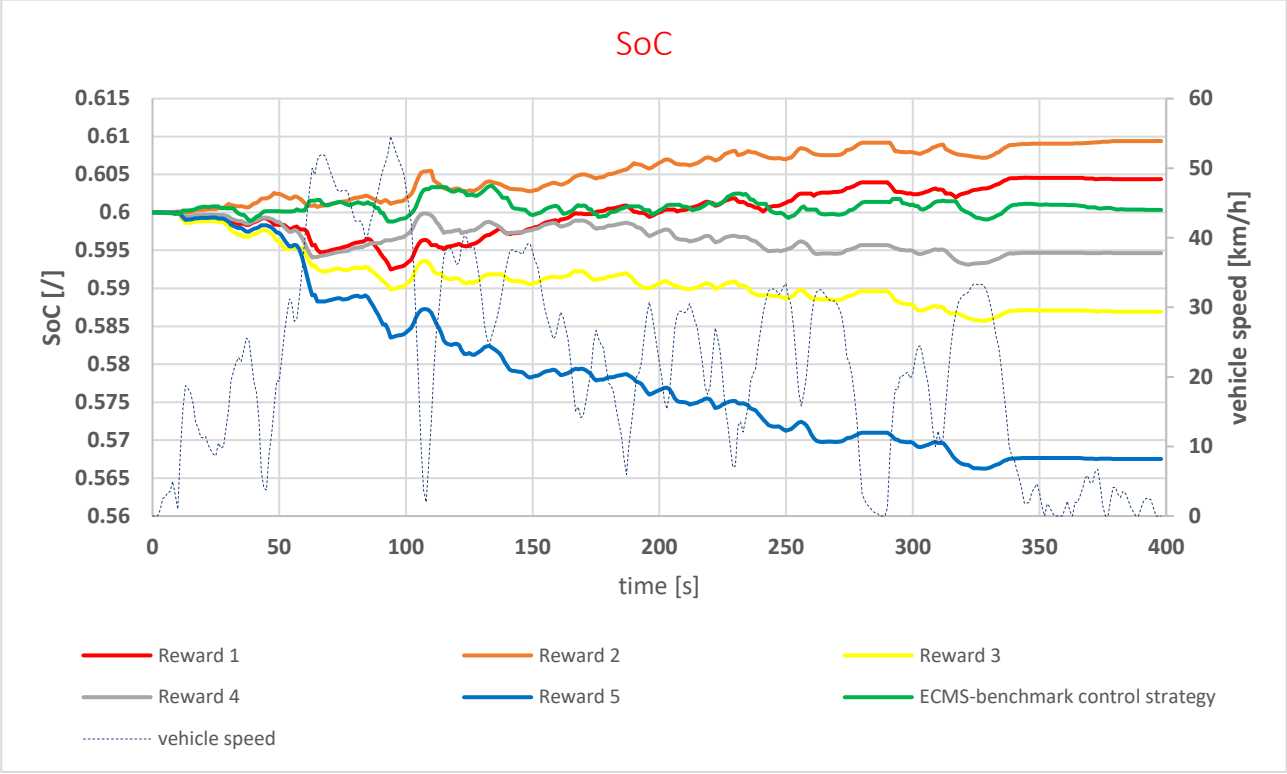


*Figure 36. SoC development (Clust 12Mod)*

The results of the last test episode in terms of state of charge of the battery are reported into *table 19*:

| Clust12Mod | |
|---|---|
| ***Reward*** | ***SoC*** |
| *Reward 1* | *0.604* |
| *Reward 2* | *0.609* |
| *Reward 3* | *0.587* |
| *Reward 4* | *0.595* |
| *Reward 5* | *0.568* |
| *ECMS-benchmark* | *0.6* |

*Table 19. SOC results in the last test episode (Clust 12Mod)*

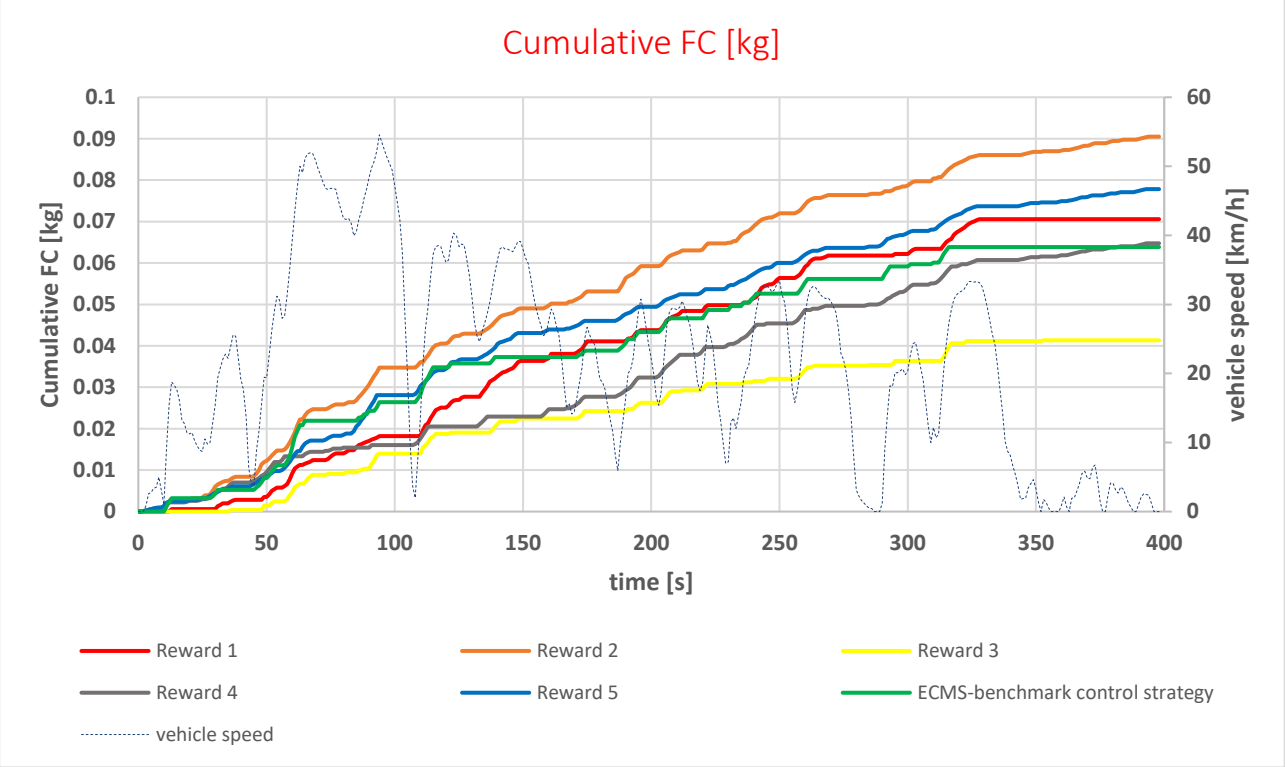


*Figure 37. Cumulative FC (Clust 12Mod)*

The results of the last test episode in terms of cumulative FC and equivalent FC are listed into *table 20* below:

| Clust12Mod | | |
|---|---|---|
| ***Reward*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** |
| *Reward 1* | *0.071* | *0.071* |
| *Reward 2* | *0.091* | *0.091* |
| *Reward 3* | *0.041* | *0.065* |
| *Reward 4* | *0.078* | *0.086* |
| *Reward 5* | *0.065* | *0.112* |
| *ECMS-benchmark* | *0.064* | *0.064* |

*Table 20. Cumulative FC and equivalent FC results in the last test episode (Clust 12Mod)*

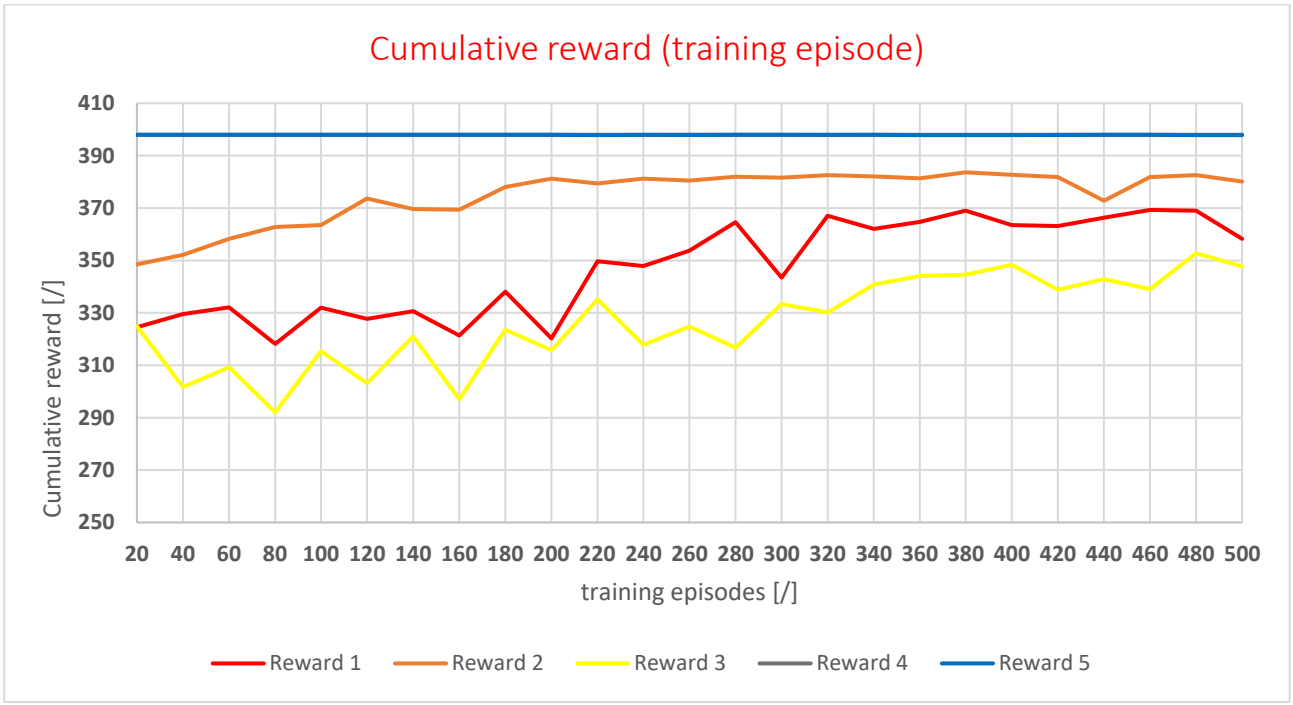


*Figure 38. Cumulative reward (Clust 12Mod)*

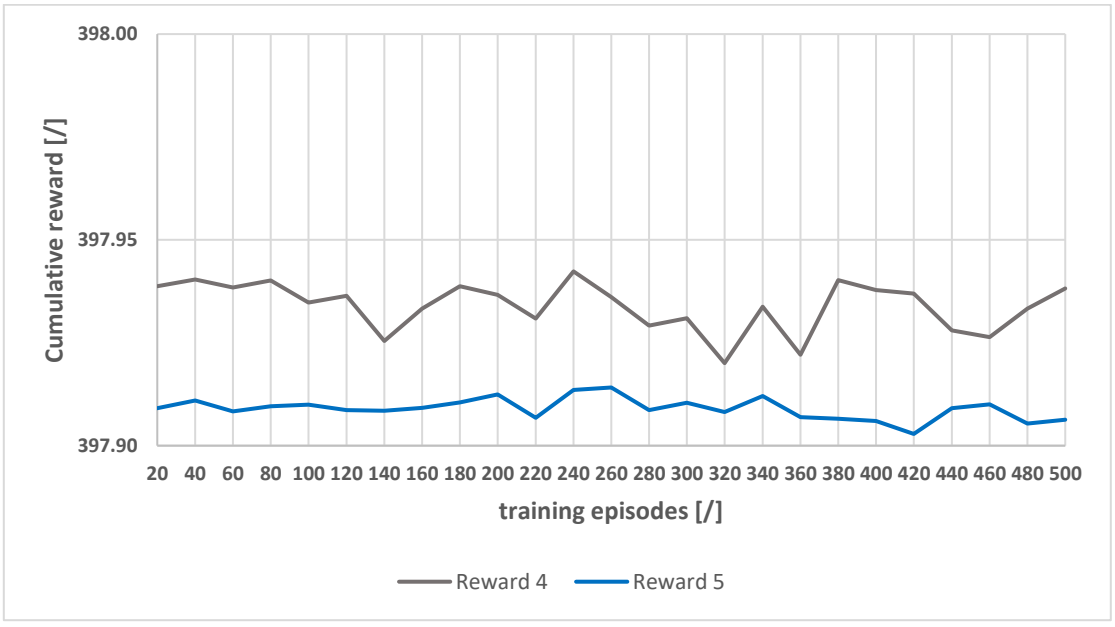


*Figure 39. Cumulative reward zoom for Reward 4 and Reward 5(Clust 12Mod)*

In the urban driving cycle, in the SoC development the Reward 1/2/3 are able to maintain the SOC level closed to the desired value 0.6. The reward-4 has a similar behaviour to the first three rewards and it is capable to maintain the SOC level close to the set reference 0.6. The reward-5 drammatically decreased the state of charge preferring to use more electric power.

In the Cumulative fuel consumption development: the reward-3 is able to reduce the fuel consumption to a value very close to the one obtained in the ECMS-benchmark. This shows how the $r_{fc,norm}$ coefficient is more efficient than $r_{fc}$.

The reward 4 and reward 5 preferring the battery usage, decreases the cumulative FC. But since in HEV energy management problem the initial SOC ($SOC_{in}$) has to be equal to the SOC at the end of driving cycle ($SOC_{end}$), this translates in increasing the equivalent FC.

The DDQN agent is stable, in fact all the training episodes are finished and the cumulative reward is increasing (*figure 38* and *39*). The agent is able to find the optimal policy in order to reduce the loss function (error).

## 6.2. Clust 11

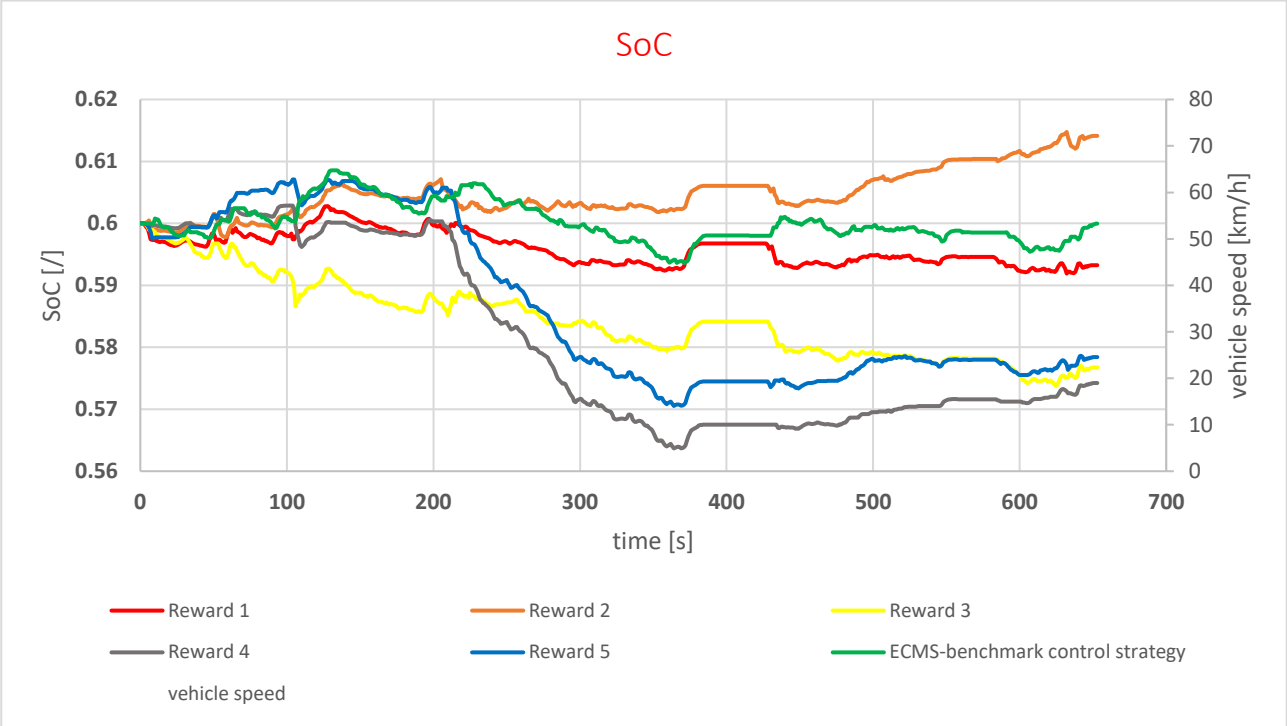


*Figure 40. SoC development (Clust 11)*

The results in terms of state charge of the battery for the last test episode are listed into *table 21*:

| Clust11 | |
|---|---|
| ***Reward*** | ***SoC*** |
| *Reward 1* | *0.593* |
| *Reward 2* | *0.614* |
| *Reward 3* | *0.577* |
| *Reward 4* | *0.574* |
| *Reward 5* | *0.578* |
| *ECMS-benchmark* | *0.6* |

*Table 21. SOC results in the last test episode (Clust 11)*

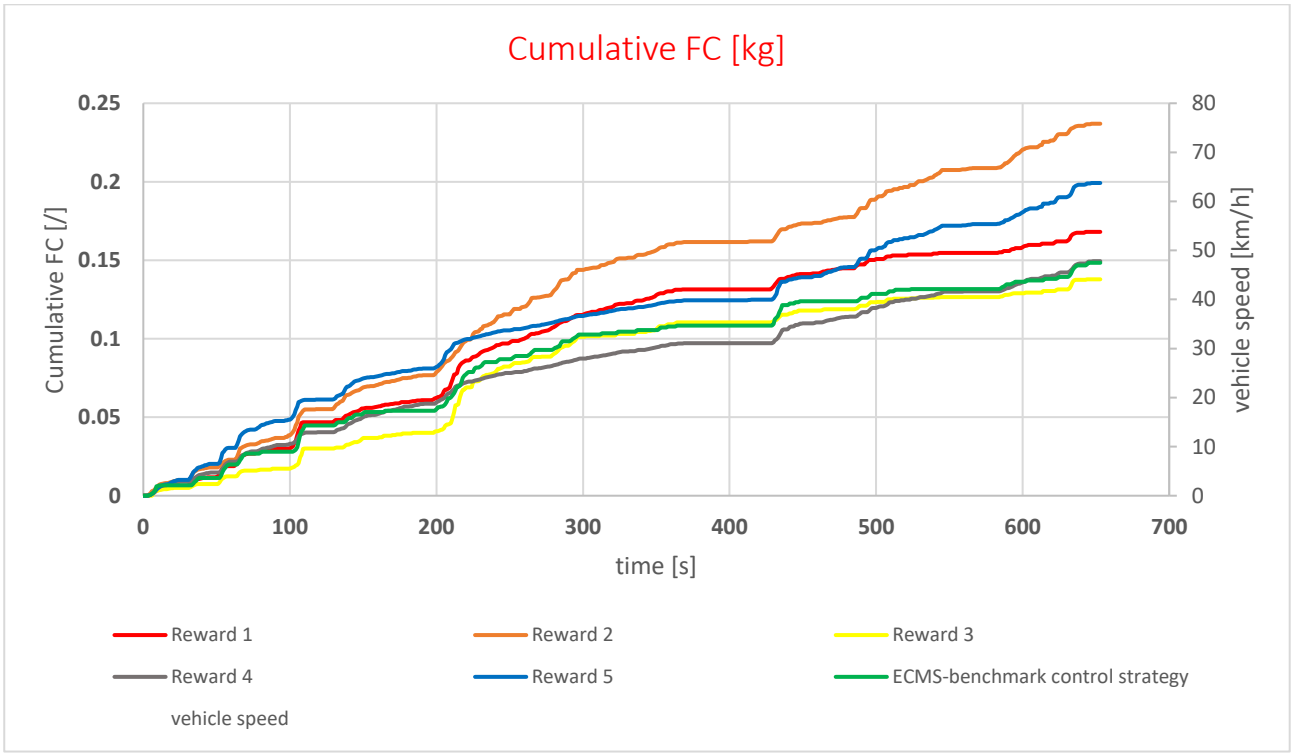


*Figure 41. Cumulative FC development (Clust 11)*

The results in terms of cumulative FC and equivalent FC in the last test episode are listed into *table 22*:

| Clust11 | | |
|---|---|---|
| ***Reward*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** |
| *Reward 1* | *0.168* | *0.178* |
| *Reward 2* | *0.237* | *0.237* |
| *Reward 3* | *0.138* | *0.171* |
| *Reward 4* | *0.149* | *0.187* |
| *Reward 5* | *0.199* | *0.23* |
| *ECMS-benchmark* | *0.145* | *0.145* |

*Table 22. Cumulative FC and equivalent FC in the last test episode (Clust 11)*

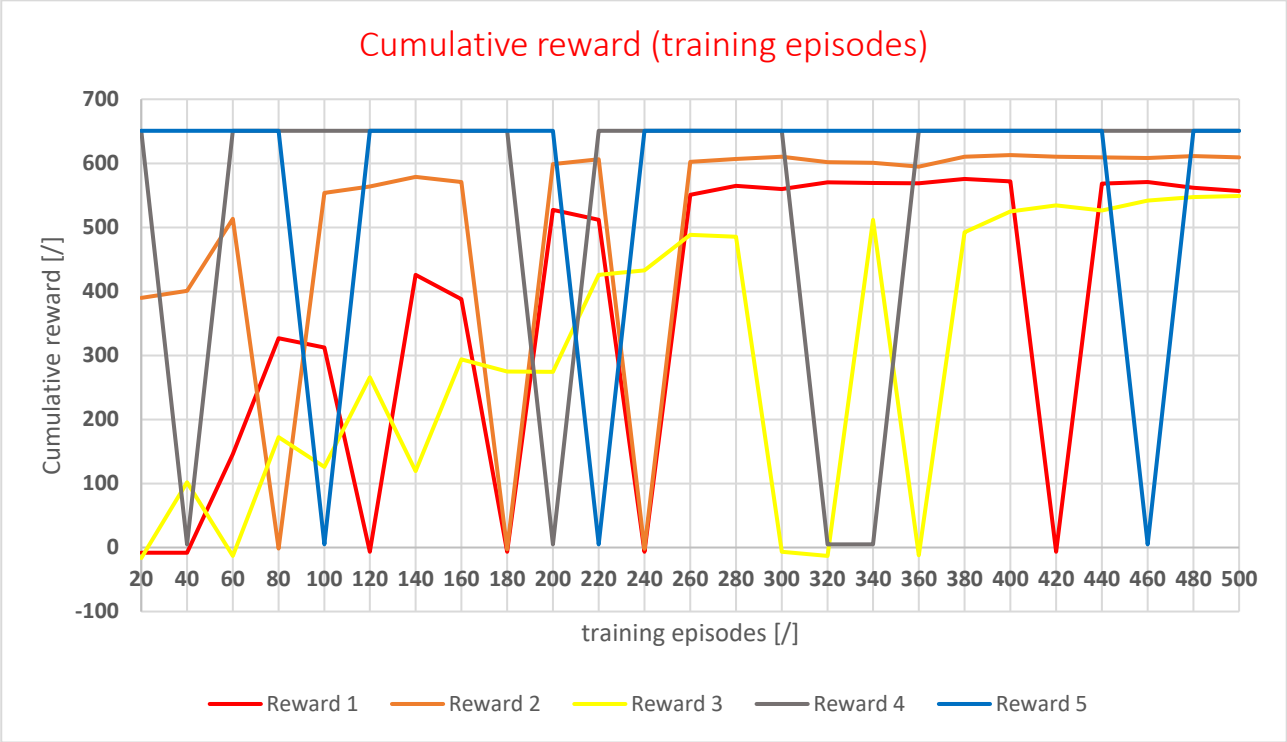


*Figure 42. Cumulative reward (Clust 11)*

In the extra-urban driving cycle, only the Reward 1 is able to maintain the SOC level close to the desired value 0.6. The reward-2 has a strange behaviour in the second part of the driving cycle increasing a lot the state of charge (excessive usage of battery charging modes). The Reward 3/4/5 instead drammatically decreased the state of charge preferring to use more electric power.

For what concerns the FC development, the reward 3 based on the weighted average is able to reduce the fuel consumption to a value very close to the one obtained in the ECMS-benchmark. This shows again how the $r_{fc,norm}$ coefficient is more efficient than $r_{fc}$. Also in this driving cycle, the reward 1 reaches good values in terms of fuel consumption rather than the reward 2.

The reward-4 and reward-5 instead have the same behaviour shown in the urban driving cycle (Clust 12Mod), increasing too much the equivalent FC respect to the Reward 1 and Reward 3.

The DDQN agent shows less stability respect to the urban driving cycle, this means that the training parameters are not well generalized changing the driving conditions (*figure 42*).

## 6.3. Clust 1

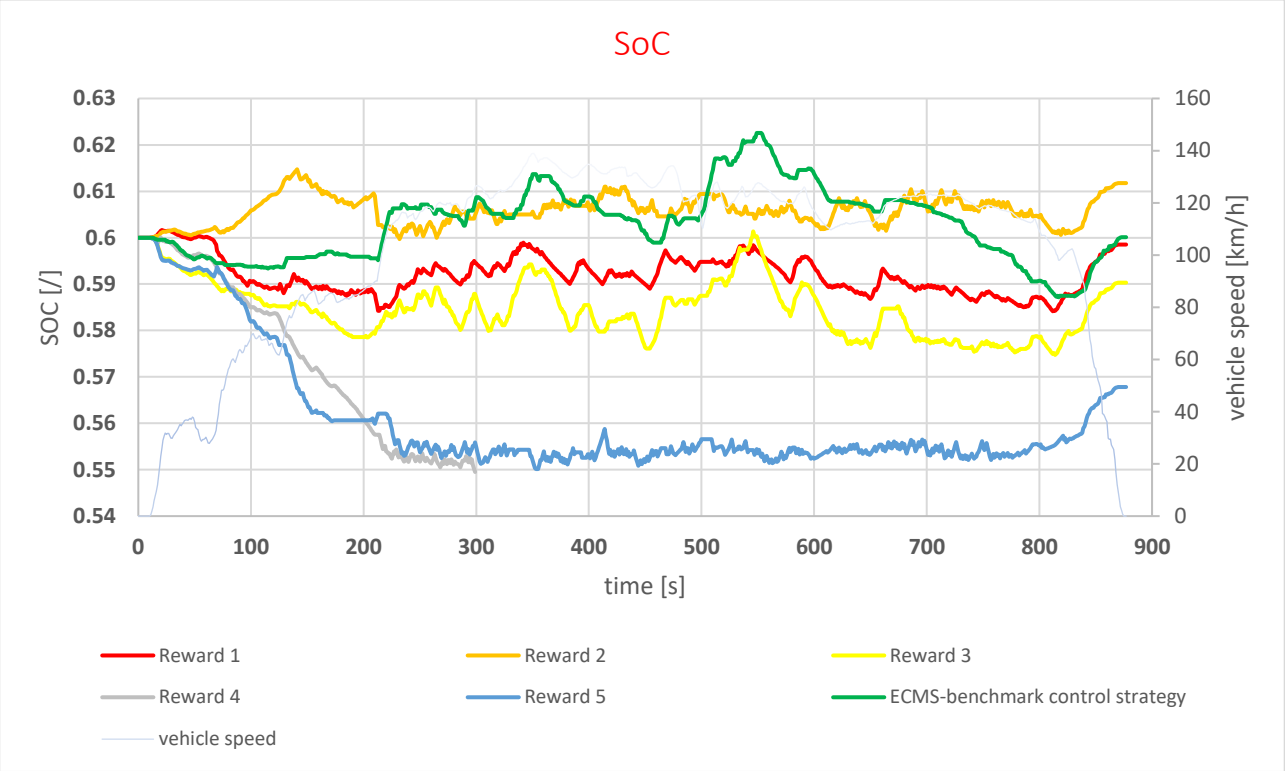


*Figure 43. SoC development (Clust 1)*

The results in terms of state of charge of the battery are listed into *table 23* below:

| Clust1 | |
|---|---|
| ***Reward*** | ***SoC*** |
| *Reward 1* | *0.599* |
| *Reward 2* | *0.612* |
| *Reward 3* | *0.59* |
| *Reward 4* | *failed* |
| *Reward 5* | *failed* |
| *ECMS-benchmark* | *0.60* |

*Table 23. SOC results in the last test episode (Clust 1)*

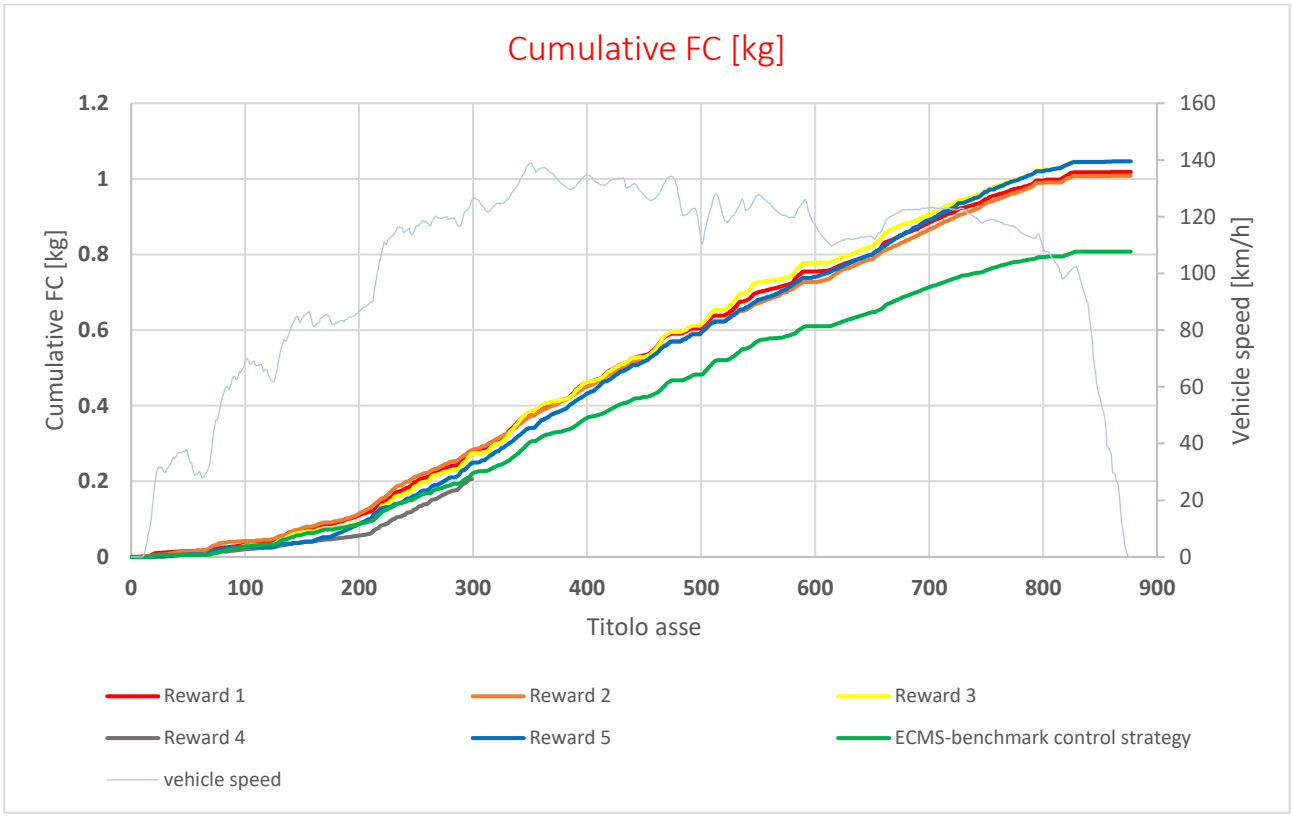


*Figure 44. Cumulative FC development (Clust 1)*

The results in terms of cumulative FC and equivalent FC are listed into *table 24* below:

| Clust1 | | |
|---|---|---|
| ***Reward*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** |
| *Reward 1* | *1.018* | *1.018* |
| *Reward 2* | *1.008* | *1.008* |
| *Reward 3* | *1.047* | *1.065* |
| *Reward 4-ECMS* | *failed* | *failed* |
| *Reward 5* | *failed* | *failed* |
| *ECMS-benchmark* | *0.807* | *0.807* |

*Table 24. Cumulative FC and equivalent FC results in the last test episode (Clust 1)*

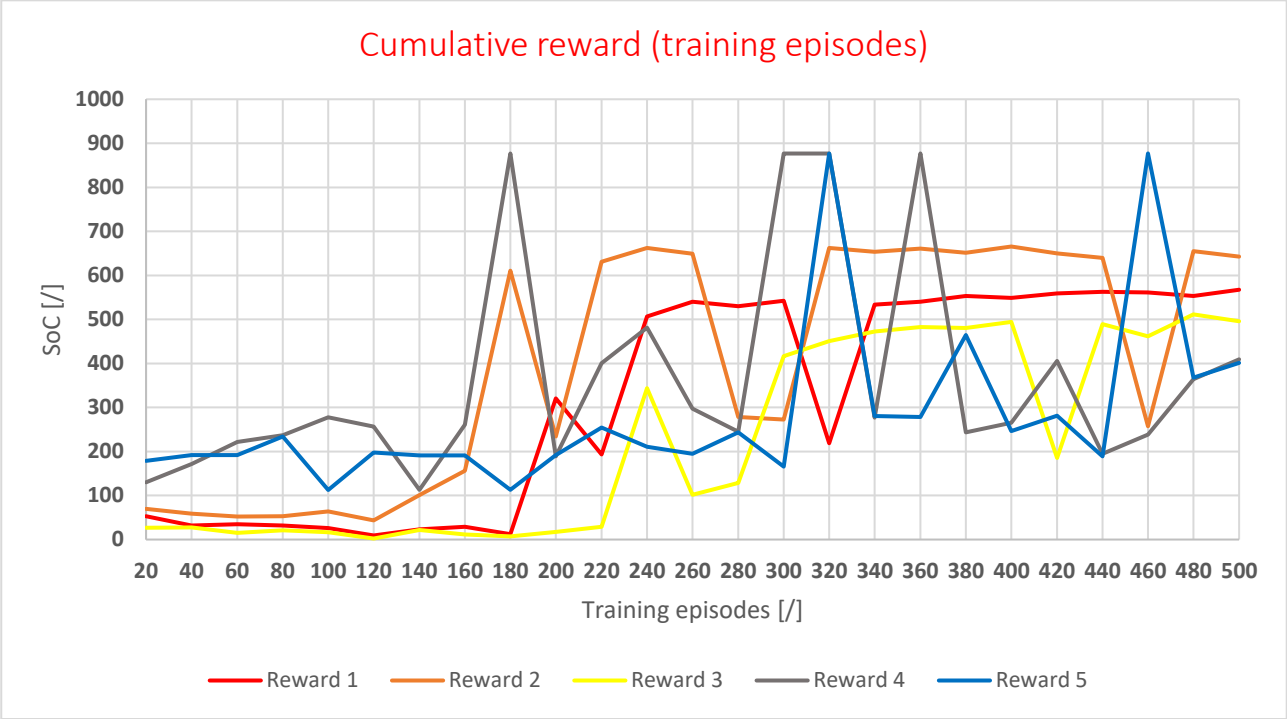


*Figura 45. Cumulative reward (Clust 1)*

In the high-way driving cycle, the reward-1 and reward-3 are able to maintain the SOC level close to the desired value 0.6. The reward-2 shows a strange behaviour during all the driving cycle increasing a lot the state of charge (excessive usage of battery charging modes) as shown in the extra-urban driving cycle (Clust 11). The reward-4 and reward-5 instead fail the last test episode since this driving cycle requires higher demanding power that decreases the SOC level below the minimum threshold causing the stop of the test episode.

For what concerns the FC development, the Reward 1/2/3 based on the weighted average can reduce the fuel consumption but far from the value obtained by the ECMS-benchmark strategy. The reward-4 and reward-5 failed.

The DDQN agent shows less stability also in the highway driving cycle, this means that the training parameters are not well generalized changing the driving conditions (*figure 45*).

## 6.4. Clust 19

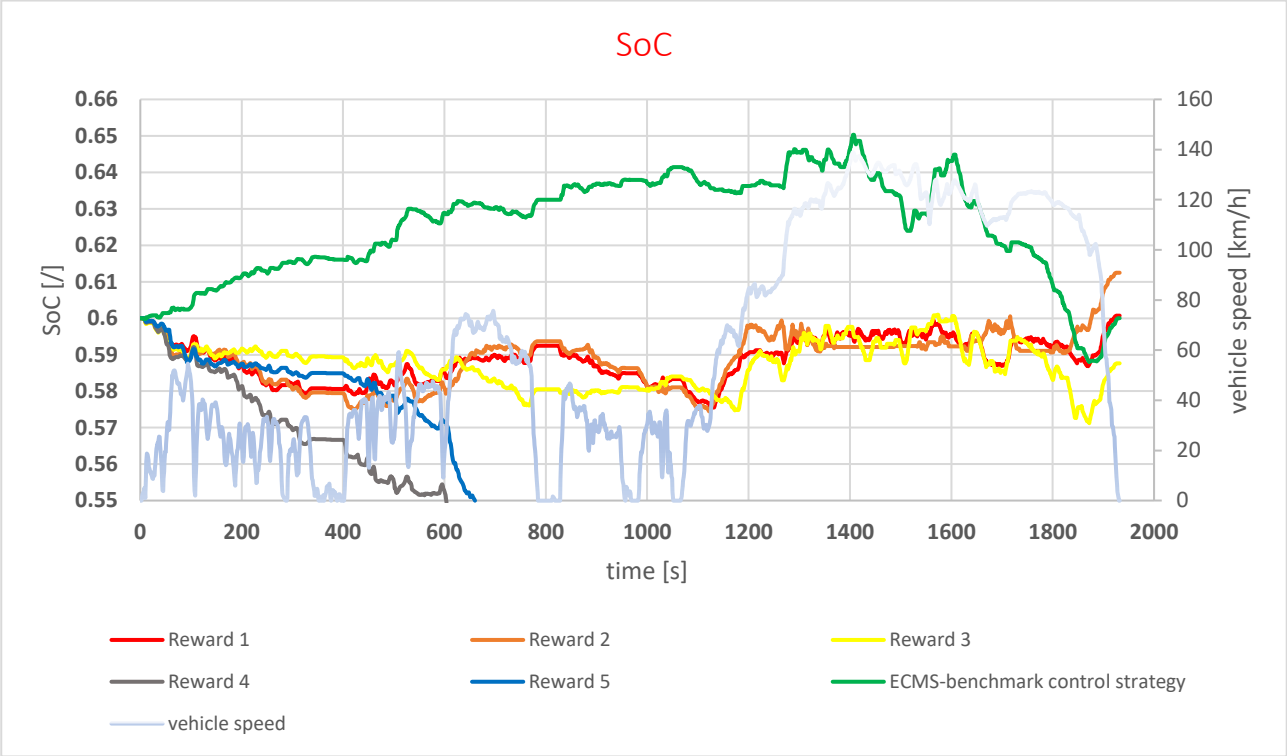


*Figure 46. SoC development (Clust 19)*

The results in terms of state of charge are listed into *table 25* below:

| Clust19 | |
|---|---|
| ***Reward*** | ***SoC*** |
| *Reward 1* | *0.601* |
| *Reward 2* | *0.612* |
| *Reward 3* | *0.588* |
| *Reward 4* | *failed* |
| *Reward 5* | *failed* |
| *ECMS-benchmark* | *0.6* |

*Table 25. SOC results in the last test episode (Clust 19)*

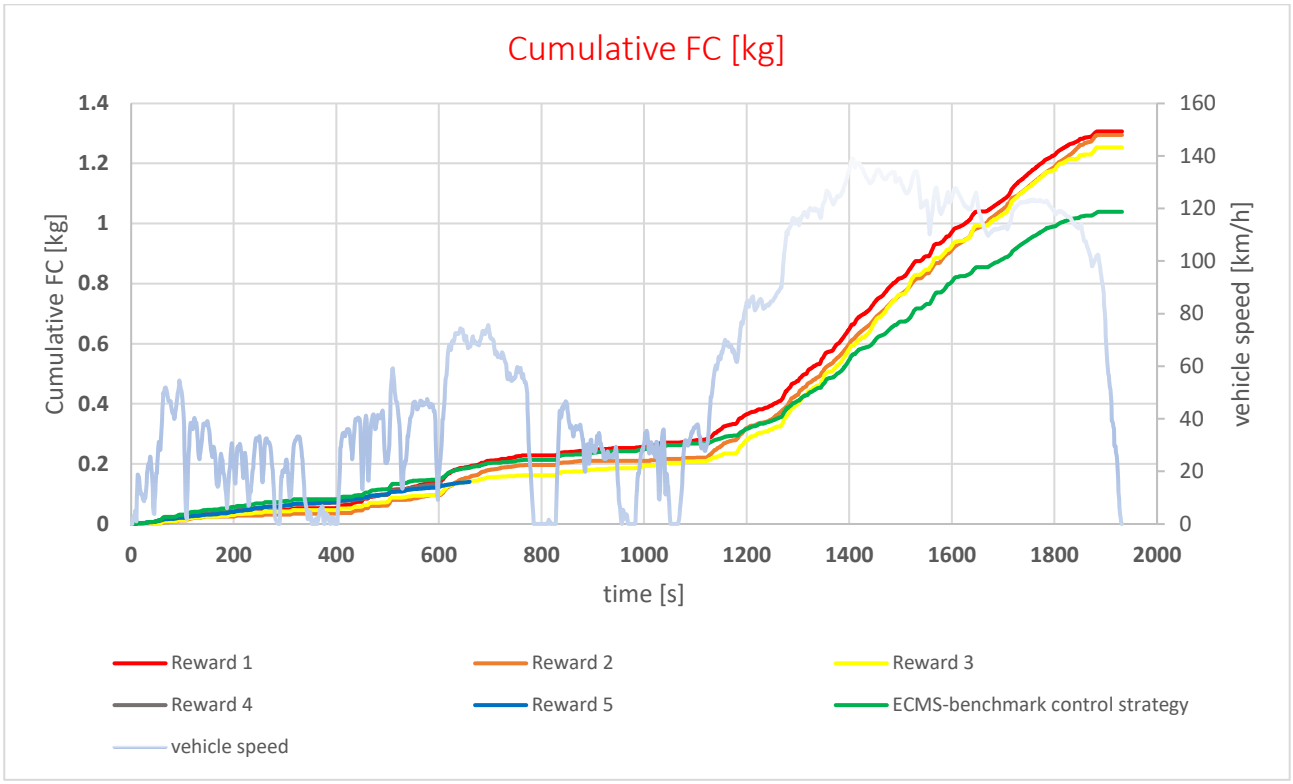


*Figure 47. Cumulative FC development (Clust 19)*

The results in terms of cumulative FC and equivalent FC are listed into *table 26* below:

| Clust19 | | |
|---|---|---|
| ***Reward*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** |
| *Reward 1* | *1.306* | *1.306* |
| *Reward 2* | *1.295* | *1.295* |
| *Reward 3* | *1.253* | *1.271* |
| *Reward 4-ECMS* | *failed* | *failed* |
| *Reward 5* | *failed* | *failed* |
| *ECMS-benchmark* | *1.039* | *1.039* |

*Table 26. Cumulative FC and equivalent FC results in the last test episode (Clust 19)*

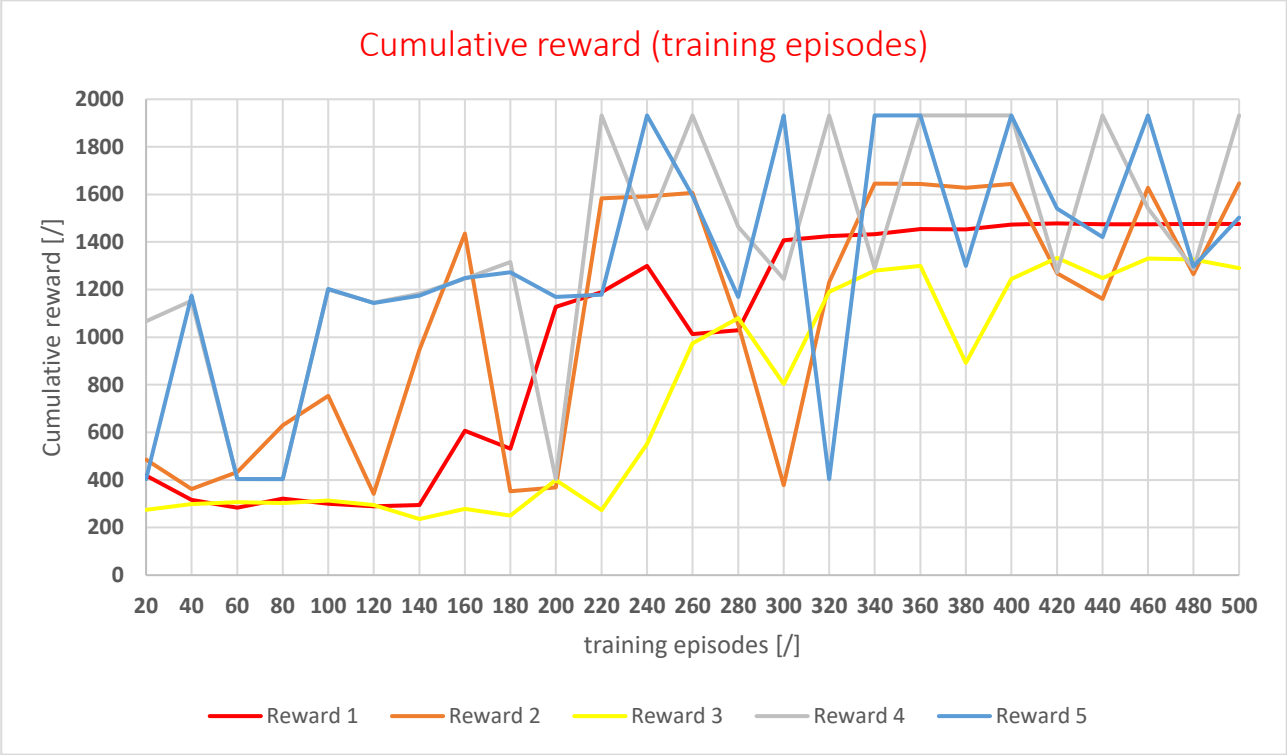


*Figura 48. Cumulative reward (Clust 19)*

.

In the mixed driving cycle, the Reward 1/2/3 based on the weighted average are able to maintain the SOC level closed to the desired value 0.6. The reward-4 fails the last test episode in the urban part of the driving cycle. The reward-5 fails the last test episode in the extra-urban part of the driving cycle.

In the FC development, the Reward 1/2/3 based on the weighted average are able to reduce the fuel consumption but far from the value obtained by the ECMS-benchmark strategy. Also in this driving cycle, the Reward-3 better reduces the fuel consumption respect the Reward 2. This shows again how the $r_{fc,norm}$ coefficient is more efficient than $r_{fc}$. The reward-4 and reward-5 failed.

The DDQN agent shows less stability, this means that the training parameters are not well generalized changing the driving conditions. Only the Reward 1 seems to be stable (*figure 48*).

## 6.5. Tables

In this section are reported the results tables for each driving cycle:

| Clust12Mod | | | | |
|---|---|---|---|---|
| ***Reward*** | ***SoC*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** | ***delta FC [%]*** |
| *Reward 1* | *0.604* | *0.071* | *0.071* | 10.55% |
| *Reward 2* | *0.609* | *0.091* | *0.091* | 41.75% |
| *Reward 3* | *0.587* | *0.041* | *0.065* | 1.81% |
| *Reward 4-ECMS* | *0.595* | *0.078* | *0.086* | 33.99% |
| *Reward 5* | *0.568* | *0.065* | *0.112* | 74.76% |
| *ECMS-benchmark* | *0.6* | *0.064* | *0.064* | / |

*Table 27. Results (Clust 12Mod)*

| Clust11 | | | | |
|---|---|---|---|---|
| ***Reward*** | ***SoC*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** | ***delta FC [%]*** |
| *Reward 1* | *0.593* | 0.168 | 0.178 | 19.97% |
| *Reward 2* | *0.614* | 0.237 | 0.237 | 59.71% |
| *Reward 3* | *0.577* | 0.138 | 0.171 | 15.25% |
| *Reward 4-ECMS* | *0.574* | 0.149 | 0.187 | 25.75% |
| *Reward 5* | *0.578* | 0.199 | 0.230 | 55.25% |
| *ECMS-benchmark* | *0.6* | 0.148 | 0.148 | / |

*Table 28. Results (Clust 11)*

| Clust1 | | | | |
|---|---|---|---|---|
| ***Reward*** | ***SoC*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** | ***delta FC [%]*** |
| *Reward 1* | *0.60* | *1.018* | *1.018* | *26.12%* |
| *Reward 2* | *0.612* | *1.008* | *1.008* | *24.86%* |
| *Reward 3* | *0.59* | *1.047* | *1.065* | *31.96%* |
| *Reward 4-ECMS* | *failed* | *failed* | *failed* | *failed* |
| *Reward 5* | *failed* | *failed* | *failed* | *failed* |
| *ECMS-benchmark* | *0.60* | *0.807* | *0.807* | / |

*Table 29. Results (Clust 1)*

| Clust19 | | | | |
|---|---|---|---|---|
| ***Reward*** | ***SoC*** | ***Cumulative FC [kg]*** | ***Equivalent FC [kg]*** | ***delta FC [%]*** |
| *Reward 1* | *0.601* | *1.306* | *1.306* | 25.74% |
| *Reward 2* | *0.612* | *1.295* | *1.295* | 24.68% |
| *Reward 3* | *0.588* | *1.253* | *1.271* | 22.37% |
| *Reward 4* | *failed* | *failed* | *failed* | *failed* |
| *Reward 5* | *failed* | *failed* | *failed* | *failed* |
| *ECMS-benchmark* | *0.6* | *1.039* | *1.039* | / |

*Table 30. Results (Clust 19)*

# 7. Conclusions

In this thesis work has been analysed an algorithm based on the Double-Deep Q-network focussing on the search of the reward function that better generalized this crucial function in Reinforcement learning problems; in order to better minimize the fuel consumption (FC) keeping the battery state of charge within the operating range for P2-HEV application. Using a sub-optimal parametric solution that allows to obtain a correct calibration of the hyperparameters of the DDQN's algorithm such as *learning starts*, *replay memory size*, *batch size*, *learning rate.* The *target update frequency*, instead, it was calibrated according to the duration of the driving cycle. The algorithm has been analysed in four different driving cycles (CLUST), that are real driving cycles that covering many possible driving scenarios for a passenger car such as sudden acceleration or deceleration, urban driving conditions such as in traffic or highway driving. The algorithm has been tested using five different reward functions: the first three based on the weighted average, the fourth and fifth reward functions are based on literature [..]. Since the fourth reward is based on equivalent consumption minimization strategy (ECMS) logic, before testing, haven been properly calibrated the equivalence factors for each driving cycle using an offline ECMS's algorithm developed in the MATLAB environment. Also in the fifth reward based on literature [..], has been calibrated the coefficient that gives a weight between the internal combustion engine fuel consumption rate and the state of charge of the battery; the calibration has been developed simulating the reward using three different combinations of the alpha coefficient. The results obtained has been compared with the ECMS algorithm developed in the MATLAB environment used as a benchmark energy management strategy. The results shown how the Reward 1 and Reward 3, based on weighted average characterized by the coefficient $r_{fc,norm}$, allows to obtain comparable results respect to the ECMS-benchmark strategy guaranteeing a very small increase of FC (*10,55%* for the Reward 1 and *1,81%* for the Reward 3) in the urban driving cycle (Clust12Mod). The Reward 2, instead, using $r_{fc}$ normalized coefficient between the fuel consumption and the maximum possible fuel consumption from internal combustion engine (ICE) dramatically increases the fuel consumption (*41,75%* in the Clust12Mod) respect to the ECMS-benchmark. It is evident how the performance of the Reward 1 and Reward 3 are reduced when a driving cycle with a more power request is used such as the Clust 1 or Clust 19. The Reward 4 and Reward 5 based on literature do not obtain competitive results in terms of fuel consumption and in addition to this, they are not capable to maintain the battery state of charge (SOC) within the operating range causing the failure of the simulation for the highway driving cycle (Clust 1) and mixed driving cycle (Clust 19).

One of the possible future studies could be a study on the loss function in order to allow a better robustness and performance of the DDQN agent on different driving scenario.

The tools used in this work could also be extended on different hybrid architecture such as plug-in HEV (PHEV) or complex HEV with more than two power sources after a suitable calibration on the parameters considering the case study.

# Bibliography

[1] Commissione europea, https://ec.europa.eu/info/policies/climate-action_it.

[2] E. Spessa, “Energy management for hybrid and electric vehicles”. Politecnico di Torino.

[3] V. Ravello, “E-powertrain components”. Politecnico di Torino.

[4] J. P. Torreglosa, P. Garcia-Triviño, D. Vera, and D. A. López-García, “Analysing the improvements of energy management systems for hybrid electric vehicles using a systematic literature review: How far are these controls from rule-based controls used in commercial vehicles?,” Appl. Sci., vol. 10, no. 23, pp. 1–25, 2020, doi: 10.3390/app10238744.

[5] Lee, H. et al. (2020). Online data-driven energy management of a hybrid electric vehicle using model-based Q-learning.

[6] Project and Development of a Reinforcement Learning Based Control Algorithm for Hybrid Electric Vehicles / Maino, Claudio; Mastropietro, Antonio; Sorrentino, Luca; Busto, Enrico; Misul, DANIELA ANNA; Spessa, Ezio. - In: APPLIED SCIENCES. - ISSN 2076-3417. - 12:2(2022), p. 812. [10.3390/app12020812].

[7] Luigi Lacaita, https://webthesis.biblio.polito.it/view/creators/Lacaita=3ALuigi=3A=3A.html

[8] Matteo Acquarone, https://webthesis.biblio.polito.it/19497/1/tesi.pdf

[9] Ahmad Taher Azar, Anis Koubaa, Nada Ali Mohamed, Habiba A. Ibrahim, Zahra Fathy Ibrahim, Muhammad Kazim, Adel Ammar, Bilel Benjdira, Alaa M. Khamis, Ibrahim A. Hameed, Gabriella Casalino, “Drone Deep Reinforcement Learning: A Review”.

[10] J. Wu, H. He, J. Peng, Y. Li, and Z. Li, “Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus,” Appl. Energy, vol. 222,no. January, pp. 799–811, 2018, doi: 10.1016/j.apenergy.2018.03.104

[11] Influence of the Reward Function on the Selection of Reinforcement Learning Agents for Hybrid Electric Vehicles Real-Time Control / Acquarone, Matteo; Maino, Claudio; Misul, DANIELA ANNA; Spessa, Ezio; Mastropietro, Antonio;Sorrentino, Luca; Busto, Enrico. - In: ENERGIES. - ISSN 1996-1073. - ELETTRONICO. - 16:6(2023), p. 2749. [10.3390/en16062749].

[12] X. Han, H. He, J. Wu, J. Peng, and Y. Li, “Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle,” Appl. Energy, vol. 254, no. August, p. 113708, 2019, doi: 10.1016/j.apenergy.2019.113708.

[13] B. Xu, D. Rathod, D. Zhang, A. Yebi, X. Zhang, X. Li, Z. Filipi, “Parametric study on reinforcement learning optimized energy management strategy for a hybrid electric vehicle,” Appl. Energy, vol. 259, no. August 2019, p. 114200, 2020, doi:10.1016/j.apenergy.2019.114200.

[14] Liessner, R., Schmitt, J., Dietermann, A. and Bäker, B. “Hyperparameter Optimization for Deep Reinforcement Learning in Vehicle Energy Management”. In Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019), pages 134-144.

[15] H. Tan, H. Zhang, J. Peng, Z. Jiang, and Y. Wu, “Energy management of hybrid electric bus based on deep reinforcement learning in continuous state and action space,” Energy Convers. Manag., vol. 195, no. January, pp. 548–560, 2019, doi: 10.1016/j.enconman.2019.05.038.

[16] Teng Liu, Xiaosong Hu, Senior Member, IEEE, Shengbo Eben Li, and Dongpu Cao, “ Reinforcement Learning Optimized Look-Ahead Energy Management of a Parallel Hybrid Electric Vehicle”.