Engineering Applications of Artificial Intelligence 146 (2025) 110135

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Explainable reinforcement learning for powertrain control engineering

C. Laflamme [a,*], J. Doppler [b], B. Palvolgyi [c], S. Dominka [b], Zs.J. Viharos [c,d], S. Haeussler [e]

[a] Fraunhofer Austria Research GmbH, Vienna, Austria
[b] Bosch Engineering, Robert Bosch AG, Vienna, Austria
[c] HUN-REN Institute for Computer Science and Control (SZTAKI), Center of Excellence of the Hungarian Academy of Sciences (MTA), Budapest, Hungary
[d] John von Neumann University, Faculty of Economics and Business, Kecskemét, Hungary
[e] Department of Information Systems, Production and Logistics Management, University of Innsbruck, Innsbruck, Austria

ARTICLE INFO

Keywords:
Reinforcement learning
Explainable artificial intelligence
Powertrain control

ABSTRACT

In this paper we demonstrate a practical post-hoc approach for explainable reinforcement learning (RL) in vehicle powertrain control. The goal is to exploit the advantages of RL yet obtain a solution that is feasible to implement in safety-critical control engineering problems. This means finding a solution that balances optimal product design with the required engineering effort, while maintaining the transparency necessary for safety-critical applications. Our method is based on initially training a neural network based RL policy and converting it into a look-up table, using a decision tree (DT) as an intermediary. The DT is limited to a certain depth, resulting in a look-up table of manageable size that can be directly tested, implemented and evaluated by control engineers. In order to evaluate this approach, a set of RL expert policies were used to train DTs with increasing depth, showing the regions where the DT solution can outperform benchmarks while still remaining small enough to translate to a manageable look-up table. Our approach involves standard Python libraries, lowering the barrier for implementation. This approach is not just relevant to powertrain control, but offers a practical approach for all regulated domains which could benefit from application of RL.

1. Introduction

The conventional approach to automotive embedded software development remains very time-consuming, necessitating a high level of expertise from software developers and control engineers (Koch et al., 2023). This is particularly true within the domain of automotive powertrain control systems where the calibration process for such systems is becoming increasingly challenging due to increasing complexity and stricter emissions regulations (Garg et al., 2023). Current control methodologies involve extensive fine-tuning of numerous look-up tables and model parameters. While Reinforcement Learning (RL) holds promise as a solution to these challenges, its adoption has been limited, with few implementations beyond prototype stages. One reason for this lack of widespread acceptance is the opacity and lack of interpretability inherent in decision-making policies generated by RL, often stored as deep neural networks. In the automotive branch this is particularly challenging, as the international standard ISO 26262 (International Organization for Standardization, 2018) regulates the software development process and this standard does not account for machine learning applications (Dominka et al., 2024; Henriksson et al., 2018; Tabani et al., 2019). As a result, the current practice continues to rely on the use of look-up tables, which, despite being inefficient to optimise, offer simplicity, transparency, require minimal computing resources when implemented and align with regulatory processes.

In this paper we address this discrepancy by considering a post-hoc method for explainable RL that exploits the benefits RL offers, yet is feasible to implement in control engineering problems. We achieve this by first optimising a neural network based policy via an RL algorithm, then train a decision tree (DT) to imitate its behaviour. This DT can then be transformed directly into a look-up table. While the look-up table immediately offers transparency and interpretability (Silva et al., 2019), the following questions remain: First, can a look-up table maintain the performance of the associated RL policy and what size does it need to be for this to be the case? Second, can the look-up table maintain the ability to generalise beyond the training data?

Here, we consider these questions for a specific powertrain control use case: the gear shifting logic for automatic transmissions. Our goal is to optimise the target gear based on system state variables using RL, but implement the final logic as a simple look-up table format. To do this, we train neural network policies, from which we train a set of DT solutions within increasing size. These DTs can be converted into look-up tables. Each table is tested and the performance is compared to the original neural network based logic, allowing us to understand

* Corresponding author.
E-mail address: catherine.laflamme@fraunhofer.at (C. Laflamme).

https://doi.org/10.1016/j.engappai.2025.110135
Received 22 May 2024; Received in revised form 10 November 2024; Accepted 20 January 2025
Available online 19 February 2025

how big the table needs to be to have comparable performance. We use a standard driving cycle (FTP 75) for training. We will test the trained RL and DTs for robustness on other standard driving cycles (NYC, HWFET, J1015, US06, UN/ECE, UDDS, WLTC) that were used in other studies (e.g., Zhou et al., 2021; Hu et al., 2023; Kerbel et al., 2023; Zhang et al., 2023). The software pipeline is built out of standard Python libraries (Gymnasium Towers et al., 2023, Stable Baselines 3 (SB3) Antonin Raffin et al., 2021, and scikit-learn Pedregosa et al., 2011) which offer stable and reliable "out of the box" algorithms. This minimises the hurdle for implementation in an industrial setting, where the goal is finding an optimal balance between increased performance and the amount of engineering effort required to obtain it. While the use case here is focused on the automotive branch, where software is regulated by the international ISO 26262-6, this approach can be used for any regulated domain, where improvements can be made through the application of RL, but standards have prohibited its direct implementation.

2. Related work

In the field of automotive control, there has been considerable work done on applying RL to a variety of different use cases, the most common being for autonomous driving (Aradi, 2022; Badr Ben Elallid et al., 2022; Kiran et al., 2022; Lindsey Kerbel et al., 2023; Tóth et al., 2024). In the case of powertrain control, RL has been applied to combustion engines, hybrid and electric vehicles to improve efficiency, reduce emissions and improve performance (for recent literature reviews on this topic see Norouzi et al., 2023; Lin et al., 2023; He et al., 2024; Louback et al., 2024).

With regards to combustion engines RL is mostly used to increase the fuel economy and decrease the emissions, where Kerbel et al. (2023) use an adaptive policy learning algorithm and Koch et al. (2023) use the Proximal Policy Optimization algorithm. Furthermore, Hu and Li (2021) develop a curiosity based exploration method for RL to control the boost pressure of a diesel engine. For hybrid vehicles, most papers use RL to optimise energy management strategies (EMS) to decrease fuel consumption and emissions and to keep the state of charge (SOC) of the battery on a predefined trajectory or reduce hydrogen consumption for fuel cell electronic vehicles. For these tasks, mainly four RL algorithms are used in literature: (i) deep Q-learning (e.g., Zhang et al., 2021; Chen et al., 2019), (ii) double deep Q-learning (e.g., Estrada et al., 2023), (iii) deep deterministic policy gradient (DDPG) algorithms (e.g., Li et al., 2019b,a; Tian et al., 2024; Tao et al., 2023) and (iv) multi agent RL algorithms (e.g., Khalatbarisoltani et al., 2022; Wang et al., 2023) are used. Finally, several papers also compare these learning algorithms with each other and develop extensions thereof (Zhou et al., 2021, 2022b; Qi et al., 2021; Hu et al., 2023).

With regards to the control of powertrain systems of purely electric vehicles, the objective of the used algorithms is to reduce energy consumption, battery loss and ageing. In Jia et al. (2020b) a Q-learning algorithm is used to train an agent for powertrain control to optimise the vehicle's speed chasing ability and power management. An imitation Q-learning algorithm is used in the study of Ye et al. (2023) where they optimise the energy management of an electric vehicle. Finally, Liang et al. (2024) and Zhang et al. (2023) analyse the performance of a DDPG algorithm to the energy management of a dual-motor powertrain of an electric bus.

While many of these works find improved performance through the application of RL, all policies are contained as neural networks. A method to make these policies explainable were not investigated. To this end, we also consider the field of Explainable AI.

Explainable AI is widely acknowledged as a crucial feature for the practical deployment of AI models, leading to many different proposed methods, in particular in the case of machine learning (Alejandro Barredo Arrieta et al., 2020). In general, these methods can be grouped into two categories: Intrinsic and Post-hoc (Andrews et al., 1995; Milani et al., 2024). While intrinsic methods develop models which themselves are directly interpretable, post-hoc methods create an additional model to explain an existing one. In the context of supervised learning, two heavily featured XAI methods are Linear Interpretable Model-agnostic Explanations (LIME; Ribeiro et al., 2016) and SHapleyAdditive exPlanations (SHAP; Lundberg et al., 2020). The former represents a model-agnostic method that creates locally interpretable explanations for single data points. By making these perturbations in the dataset, the importance of a feature can be deduced, as was used by Munkhdalai et al. (2019) to explain neural network models in credit scoring applications. Shapley values measure how much an individual feature in the neural net affects the output. For instance, SHAP has been used to interpret ensemble machine learning models in identifying risk factors during general anaesthesia (Lundberg et al., 2018) and in predicting driver fatigue, revealing hidden patterns between fatigue levels and various physiological measures in the used machine learning model (Zhou et al., 2022a). These explanations not only highlight key variables in the model but also enhance user trust in practical applications (Ayoub et al., 2021). Compared to LIME, SHAP provides superior explanations, particularly in terms of local accuracy and consistency, making it a more reliable choice for interpreting machine learning models (Lundberg et al., 2020).

While SHAP and LIME can help provide insight into the feature importance in a supervised learning context, they cannot be used to extract an explainable model that can be implemented in the place of e.g., a neural network. To this end, several approaches have been used to extract rules from a trained neural network. In Towell and Shavlik (1993) an analytical method is developed to extract rules from weights of the neural network. In Lal and Mithal (2022) an algorithm NN2Rules is developed which converts a trained neural network into a rule list. In Müller et al. (2022) a graph neural network is trained using gradient descent with multi-layer perceptrons. Then, each perceptron is replaced with a decision tree while keeping the original GNN message passing structure. In Jia et al. (2020a) a convolutional neural network is decomposed into a feature extractor and a classifier, and a DT is extracted from only the classifier.

One main challenge in many XAI applications is the trade-off between interpretability and performance, relevant in any use case where performance is coupled with model complexity (Arrieta et al., 2020). What determines this trade-off, however is dependent on the end-users who can accurately analyse the potential risks of, e.g., misclassification with an interpretable model (Ahmad et al., 2018).

The subfield of explainable RL is focused on gaining insight into the agent's decision-making process. While this has been done with many different techniques,[1] including fuzzy control systems (Bautista-Montesano et al., 2020), neuro-fuzzy solutions (Viharos and Kis, 2015) and programmatic policies (Verma et al., 2018), a very common method, and the one focused on here, is DTs, as they epitomise interpretability (Wan et al., 2020). In Roth et al. (2019) the algorithm Conservative Q-Improvement is introduced, which results in a DT which represents the Q-values directly. This method is additive, and is not focused on reducing the number of total nodes. In Silva et al. (2020), the authors use a differentiable DT as the function approximator for a policy gradient algorithm. In this work tests are done on two test environments and a user study is conducted to evaluate the interpretability of the resulting DTs.

In addition to methods which train the DT directly, several methods are based on imitation learning — learning from an expert policy. For example, Bastani et al. (2018) introduces the VIPER algorithm which learns DTs based on the imitation learning algorithm, DAGGER (Ross et al., 2011). It extends this algorithm by training a sequence of DTs on data points sampled based on how "critical" they are measured to

[1] Please see Vouros (2022) and Hickling et al. (2023) or Milani et al. (2024) for recent literature reviews on this topic.

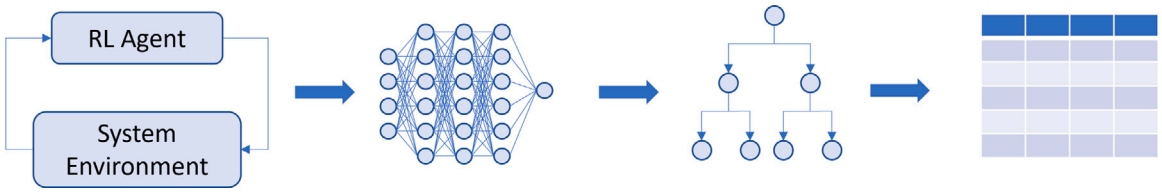

Fig. 1. A schematic of the method used. An RL agent is trained by interacting with the system environment, resulting in an optimised neural network based policy. From this neural network, a DT is trained with a fixed, maximum number of nodes. Finally, a look-up table is generated directly from the DT.

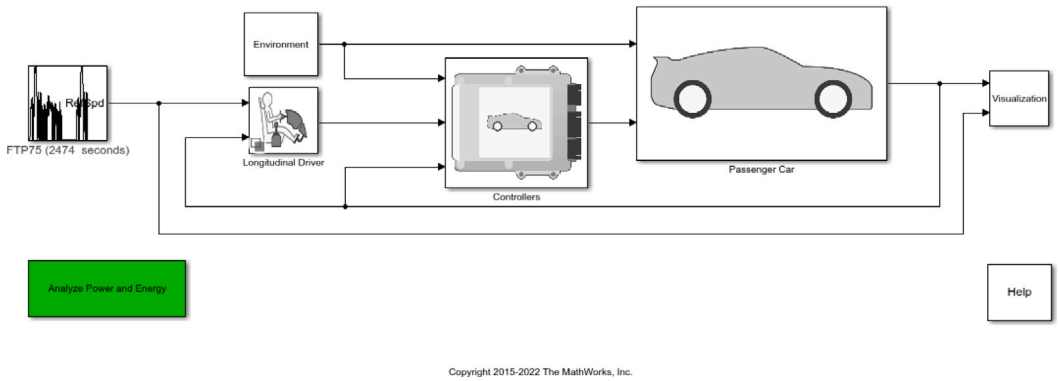

Fig. 2. Matlab conventional vehicle reference application.

be. In Vinícius G. Costa et al. (2024), the authors go a step further and use an evolutionary approach to evolve the DTs resulting from imitation learning with a fitness metric that prioritises interpretability and consistent high performance. In Coppens et al. (2019) soft decision trees are distilled from RL policies which have been originally trained with a PPO algorithm. Soft decision trees (Frosst and Hinton, 2017) are hybrids between NN and DTs and thus are not able to be implemented in the control software directly as a look-up table. Zhang et al. (2020) use genetic programming to extract control policy from the neural network, including a method to simplify the resulting policy. Finally, Dai et al. (2022) extracts oblique decision trees (Yang et al., 2019) from data generated from the control policy of the agent for power system emergency control where they argue it could be directly implemented in edge devices where computing resources are limited. Another, continuously growing, dynamic Q-table based RL method is given in Viharos and Jakab (2021).

Although there is a substantial body of research in XAI, practical methods to implement RL in safety-critical contexts remain underexplored. Our approach addresses this gap with three key points that distinguish it from previous work: First, we convert the RL policy into a lookup table, enabling straightforward integration into control software. Second, our method is designed to be implemented entirely using standard python libraries, reducing the complexity of adoption. Finally, we assess and quantify the practical tradeoff between performance and interpretability, providing a balanced approach tailored for real-world applications.

3. Methods

The method used in this paper has three steps: First, train a neural network based policy using an RL algorithm. Second, train a DT based on this (optimised) policy by sampling the policy in the state space. Finally, we translate the resulting DT (which has been limited to a given number of parameters) into a look-up table.

This three step process is shown in Fig. 1. Here, we demonstrate these steps for a powertrain control use case: Optimising the gear shifting logic for an automatic transmission. This particular use case is chosen as it is a well established problem, regulated with various standards and allows us to focus on a feasibility study of a real life application. The goal is that this demonstration can then be applied to newer, less established and higher risk use cases. In this gear shifting use case, the target gear is selected based on system state variables: engine speed, accelerator pedal position, which correlates to the desired engine torque, and velocity of the vehicle.

We note that we initially attempted to train a Q-table solution directly with a Q-learning algorithm. However, this approach suffers several major drawbacks. One, the sampling required for a table to converge is greater than for a neural network, as each table entry must converge independently. Second, the discretisation of the Q-table must be done by hand in advance. This, coupled with the long convergence times means that a manual optimisation of the discretisation is not optimal. Lastly, the Q-table lacks the ability to generalise to situations that are not seen in training.

3.1. Simulation

The use case is implemented with a modified version of the "Conventional Vehicle Reference Application" simulation which is part of the Simulink Powertrain Blockset library (Simulink, 2024a). These simulations, created by MathWorks, are an accurate representation of the models used in the software engineering process in practice. In the simulation a driver controls the acceleration and brake pedals of the vehicle, trying to follow a given velocity curve. Based on the selected gear, the accelerator pedal position determines the resulting engine torque, which ultimately leads to a certain engine speed and vehicle speed. As implemented in Matlab, the control unit has a shifting map which contains the logic of which gear is selected (based on the accelerator pedal position, current gear and current vehicle speed) and has been optimised for fuel economy (Simulink, 2024b). The simulation, including all input and output variables is shown in Fig. 2.

Our goal is to improve this control unit using RL, without having to directly implement a neural network (where the policy is originally stored). Rather our goal is to implement the policy via a look-up table which has been trained to mimic its behaviour (but with fewer parameters). The optimised Matlab control unit logic is used as the benchmark.

In order to take advantage of the benefits of Python, which includes many standard libraries for RL, the Matlab simulation is compiled to

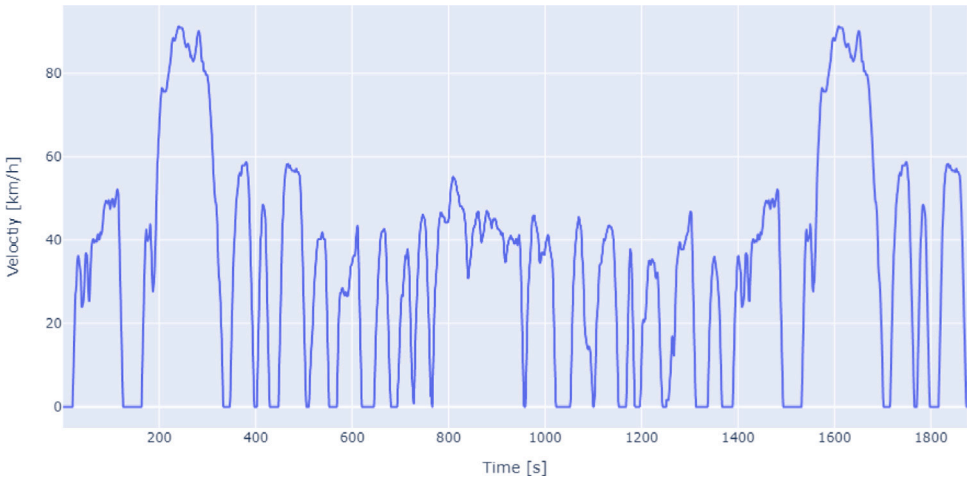

Fig. 3. The Federal Test Procedure 75 (United States Environmental Protection Agency, 2023).

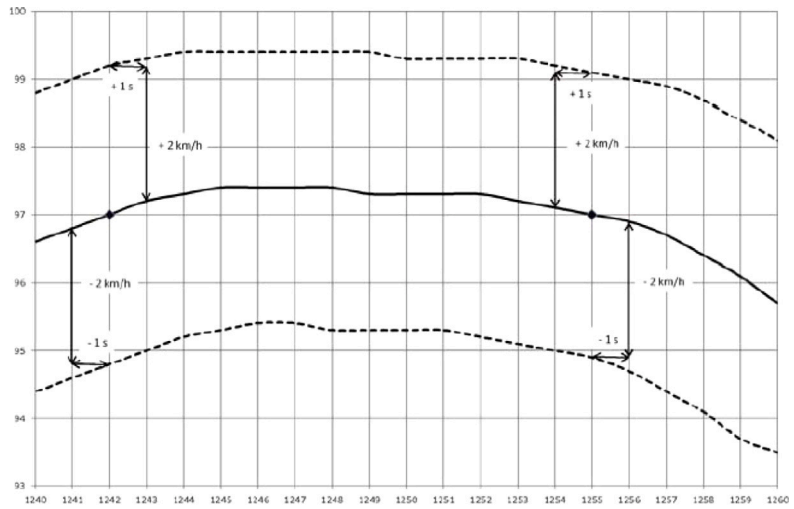

Fig. 4. Speed trace tolerance. Upper limit: 2 km/h higher than the highest point of the trace within ±1 s of the given point in time; Lower limit: 2 km/h lower than the lowest point of the trace within ±1 s of the given time. Deviations that exceed 1s are defined as one speed fault. No more than ten faults per test cycles are tolerated.
Source: Figure taken directly from The European Commission (2017).

generate a shared library .dll (using the Embedded Coder toolbox) which can be loaded and carried out stepwise in Python, where, at each step, the state parameters are given as output and the new gear can be given as an input (at a specific time point by a specific driving cycle). A given policy is evaluated based on two properties: One, the final fuel economy (the ratio of the integrated distance and fuel consumption throughout the entire cycle) and two, the ability of the vehicle to follow the given reference velocity curve. This reference velocity curve is given to the simulation as input, and can be chosen as one of various standardised driving cycles. The driving cycle used for training is the "EPA Federal Test Procedure" cycle, shown in Fig. 3.

3.2. Reinforcement learning implementation

Given the simulation as described in the previous subsection, we now set it up for a practical implementation of RL. Using the shared library .dll as the simulation base, we integrate it in a gymnasium environment (Towers et al., 2023) which ensures the structure for the application of RL, and the ability to use standard algorithms from e.g., SB3 (Antonin Raffin et al., 2021). In our simulation the state space is defined by three simulation parameters: Vehicle speed, engine speed and accelerator pedal position, and the action space is the discrete space of possible gears ($[0, 8]$). Our reward at each RL update step $R(t)$ is defined as a combination of the fuel flow (integrated from the last RL update step, which in our implementation is much larger than the simulation step-size) and the absolute deviation between the reference velocity and the actual vehicle velocity:

$$R(t) = -\alpha |v_{\text{ref}}(t) - v_{\text{act}}(t)| - \sum_{\tilde{t}=t-\Delta t}^{\tilde{t}=t} f(\tilde{t}), \tag{1}$$

where $\Delta t = 0.5$ s is the time since the last RL update (larger than the simulation step size of 2 ms), $v_{\text{ref/act}}(t)$ is the reference/actual vehicle velocity, respectively, at time t, and $f(t)$ is the amount of fuel intake per second at time t. The parameter α represents the weight between the two reward factors and is a free parameter.

While the reward function $R(t)$ is used for training, for the final evaluation of trained policies we use two standard measures: the total fuel economy and the number of faults, as defined by the Worldwide Harmonised Light Test Procedure (WLTP) (The European Commission, 2017). The first measure, the fuel economy, is defined as the distance travelled divided by the fuel intake over the entire driving cycle, and given in miles per gallon.[2] The second measure, the number of WLTP

[2] The choice of non metric units is due to its use in e.g., the Matlab evaluation.

Table 1
Summary of RL experiment parameters, where α is the reward weight as defined in Eq. (1), $N_{episodes}$ is the number of episodes used for training, and the other parameters are those of the SB3 implementation of the PPO algorithm. Hyper-parameter optimisation is done within the given parameter range using the software Weights and Biases (Biewald, 2020).

Parameter	Value
Simulation step size	2 ms
RL update step size	500 ms
Algorithm	PPO (SB3)
Batch size	{16,32,64,128}
N steps	{512,1024,2048,3072}
Learning rate (lr)	$[10^{-5}, 10^{-3}]$
α	[0.1, 1]
γ	[0.5,0.99]
$N_{episodes}$	50

faults, is a measure of the velocity following. While the ability to follow the velocity curve can be measured in a number of different ways, the WLTP tests verifies if the vehicle speed is outside of the allowable range for a given reference time. If so, a fault condition is set. For the WLTP, the fault conditions are given in Fig. 4. While the number of faults is the standard way of measuring the velocity following, the absolute deviation of the velocity curve is part of the reward function given in Eq. (1), as it includes more information, and leads to more efficient learning.

While this reward function takes into account two technical aspects (the fuel economy and the ability to follow the reference velocity curve) it does not include aspects that impact the driver's comfort (e.g., noise from high engine speeds) or that account for vehicle wear-and-tear (e.g., high frequency of shifting). Quantifying these aspects, and including them in the reward function would lead to a more balanced policy and is the subject of future work.

With the state space, action space and reward defined, the environment is then used for training, employing the PPO algorithm as implemented in SB3 (Antonin Raffin et al., 2021) and tuning hyper-parameters using the program Weights and Biases (Biewald, 2020). For training, the FTP cycle (see Fig. 3) is used, as this is the cycle used for training the benchmark policy. A summary of the RL experiment parameters is given in Table 1.

Due to the high redundancy of the solution space (even for a fixed value of α, by trading velocity following for fuel economy a new solution can be found with similar reward) many different (optimal) policies are found, each with a slightly different trade off between fuel economy and velocity following. The parameter α is limited to a range of values ([0.1,1]) which ensures the optimal policies have a better fuel economy than the benchmark. At this point, if neural networks could be implemented easily in the control software, any of these solutions could be implemented for improved performance from the benchmark. However, the goal of this paper is to consider how we implement an explainable, rule-based policy, which benefits from the RL policy. To this end, the set of trained optimal policies is then used for the further steps.

3.3. Decision tree

Using the set of trained, neural-network based policies obtained from the PPO algorithm, our next step is to derive a DT, which mimics the policy of the neural network. If the performance of the DT is sufficient, then it has the advantage that it can be easily translated into a look-up table which can be directly implemented in embedded control units. This ensures the explainability of the safety-critical control and allows decisions to be tracked and justified directly. Given the size of our state space, in order to create the look-up table we begin by sampling the entire state space uniformly to create a static data set from the policy. This sampling is done uniformly as the state space is small enough for this to be feasible, and uniform sampling helps avoid overfitting to the training data (e.g., as would be the case if we sampled only points that were visited during training). In cases where this is no longer feasible due to the state space dimension, different methods are available to sample the state space (Bastani et al., 2018; Ross et al., 2011). Then, based on this data set, a decision tree is trained using standard supervised learning algorithms given in the Python library scikit-learn (Pedregosa et al., 2011). The DT depth is fixed, to obtain a set of DTs with increasing size for each RL agent. Note that this training is done with the goal of maximising the accuracy of reproducing the dataset, however the decision tree policies will be evaluated by their fuel economy and number of faults when implemented directly in the simulation.

3.4. Look-up table

Finally, once a DT is obtained which imitates the behaviour of the RL policy, it can be translated into a look-up table. Look-up tables have the advantage that they are used ubiquitously in control engineering problems, they are immediately transparent and once implemented can be verified through expert knowledge, building trust of the policy for the responsible engineers.

The look-up table is built from the DT in the following way: For each node of the DT, a decision is made based on a threshold of one state space variable. By considering every node, a set of thresholds for each variable is obtained. The look-up table is then built out of these threshold values and is filled with the values given in the final leaves of the DT. The resulting size (number of entries) of the look-up table is given by

$$S = \prod_{i=1}^{M} (N_i + 1) \tag{2}$$

where S is the resulting size of the table, M is the number of dimensions of the state space and N_i is the number of thresholds for that dimension. Thus, the number of entries of the look-up table increases with the depth of the DT, and both measures can be used as a quantitative measure of interpretability and explainability (Breiman et al., 1984; Guidotti et al., 2018). While many approaches exist for quantifying interpretability (see Milani et al. (2024) for a comprehensive review) the number of entries of the look-up table gives us a measure encapsulating the points important for direct implementation. Specifically, in this study, we focus on two critical aspects: the ease of software implementation, which is directly linked to table size, and the ease with which a test engineer can review the table. Although there is no strict threshold for when a look-up table becomes non-explainable, each use case requires a balance between performance and explainability, as demonstrated in the gear-shifting example below.

4. Results and discussion

4.1. RL agent solutions

We begin by training a set of RL agents, and these results are shown in Fig. 5, which highlights the trade-off between fuel economy and the number of faults. Each of these solutions represents a neural network based policy that outperforms the benchmark policy either with significant improvement in the fuel economy while maintaining the same number of velocity faults or a slightly better fuel economy with no velocity faults. If implementing a neural network policy into control software were realistic, then the task would be finished. The RL solution could be chosen to fit the trade-off between fuel economy and velocity faults desired (e.g., a higher fuel economy solution could be implemented as "eco" mode, or the lower fuel economy solution as "sport" mode) and implement that policy directly. However, as a neural network based policy is prohibited by the automobile software regulations, here our focus is deriving a look-up table from these policies, which is done in the following sections.

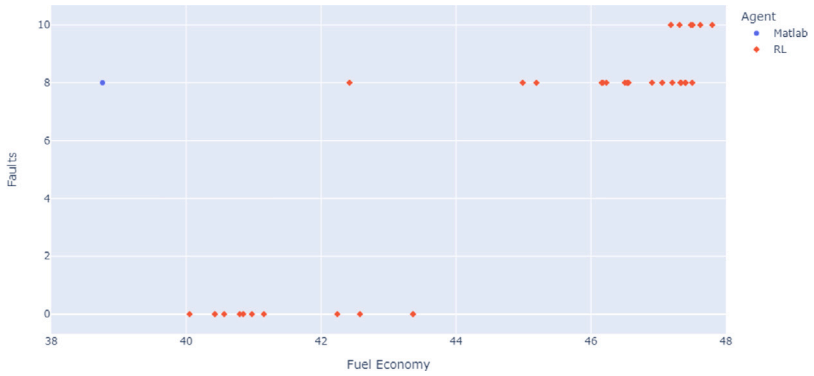

Fig. 5. Fuel economy and number of faults for different RL agent solutions. Results are compared to the Matlab benchmark solution.

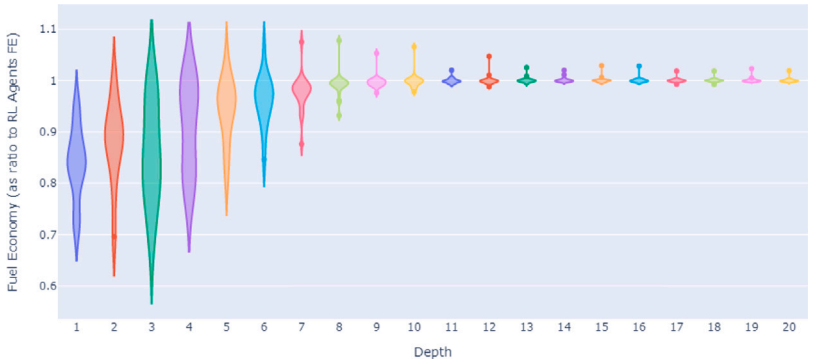

Fig. 6. Violin plot of the fuel economy of the DT with a given depth, given as the ratio to the fuel economy of the associated neural network policy. Note that ratios > 1 are possible, as the error of the DT can, by chance, increase the performance. A set of 34 RL neural network policies are used, resulting in 34 different DT policies for each depth. Note that DT policies with > 10 faults are excluded from the plot, as their fuel economy can then be misleading. The complete set of results, including those with > 10 faults is given in the supplementary material.

4.2. Decision tree solutions

In order to learn from the policy of the neural network, we create a sampled data set of state action pairs. This sampling is done with uniform sampling in the (sufficiently small) state space with 100 points taken for each dimension of the state space. For larger state spaces where this is no longer feasible a more efficient sampling strategy must be used (Bastani et al., 2018; Ross et al., 2011). A more in-depth analysis of different sampling and cross-validation for the DTs is given in the supplementary material. If the neural network is sampled sufficiently, and the DT has a sufficient number of nodes (comparable to the number of parameters of the neural network) it is clear that the DT will reproduce the results that the neural network gives. However, such a large DT would offer no benefits in explainability, and with so many parameters, it would not be realistic to implement it as a look-up table. Rather, the goal is to find the smallest possible DT for a given performance. To this end, for each RL policy we train a set of DTs, each with a different set depth.

The resulting policies are then tested independently on the FTP cycle, and the resulting fuel economy of each test run is shown in Fig. 6. The fuel economy is given as the ratio to the fuel economy resulting from implementing the associated neural network policy, allowing the different policies to be compared to one another. In this plot we can see the convergence of the DT policies to that of the RL policy with a depth of around 9. Smaller decision trees do have cases where the performance is comparable (or exceeding) to the RL policy. These well performing models could be selected, but one needs to be aware of the higher variance within these trained solutions.

This figure shows the tradeoff between performance and interpretability directly. While the deeper trees have a more stable performance when compared to the optimal RL policy, their higher depth necessarily means they will translate to a look-up table with a higher number of entries. Each use case needs to be considered individually to decide where the cutoff between performance and interpretability lies, depending on the particular requirements.

4.3. Evaluation on all driving cycles

In the previous section the evaluation of the RL and DT policies was done on the FTP cycle, which was also used for the training of the RL policy. In this section, we evaluate how well these policies can generalise to independent driving cycles, without any further training. These further cycles are shown in Table 2 and are seen as the standard cycles for automobile design. While these cycles have been constructed to mimic real-world conditions as well as possible (Ericsson, 2001; Fontaras et al., 2017), a final test of how well the policies perform in real-world conditions must be done be implementing them in real vehicles, which is the subject of future work.

The results of the benchmark policy, each RL policy and their associated DTs (with depth 3 and 6) are shown in Fig. 7. There, we see that the DT behaviour mimics the general behaviour of the RL agent closely. The smaller DT has generally a lower fuel economy and higher variance, indicating that a smaller DT is not capable of generalising to other driving cycles as well as a larger DT or the original neural network policy. However, by DT of depth 6, it seems that the generalisability mimics that of the RL policy. We also see that the two driving cycles that are most different to that used in training (i.e., US06 and WLTC class 3, which focus on high speed driving, vs FTP which is focused on accelerating and decelerating) have the poorest performance, as would be expected.

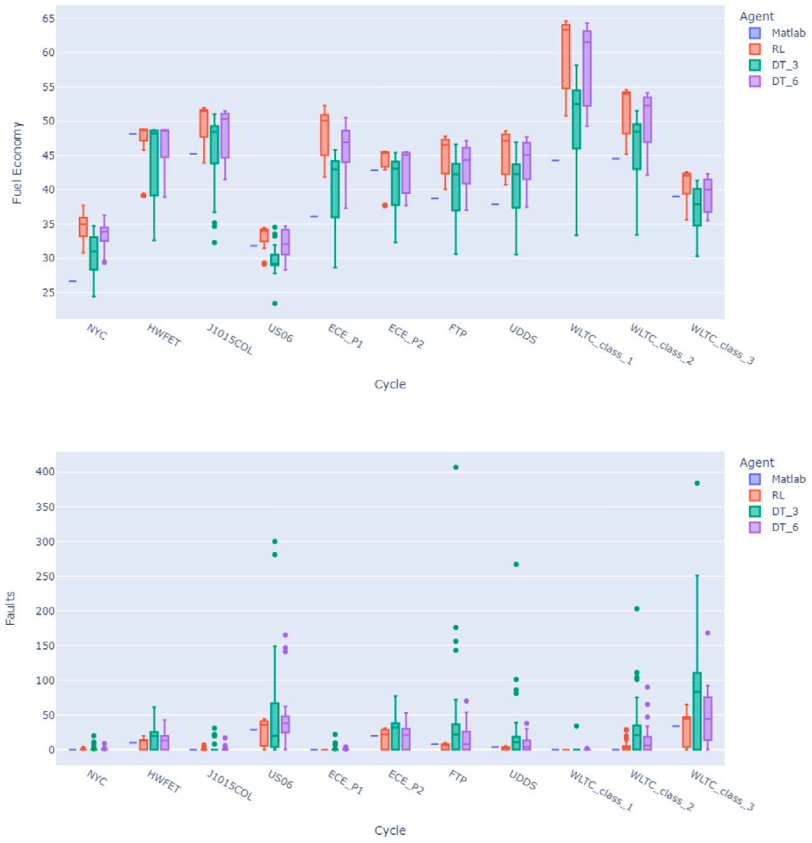

Fig. 7. *Above:* The FE of the Matlab benchmark, the set of RL Agents and their associated DTs of depth 3 and 6 when tested on the complete set of driving cycles. All policies have been trained on the FTP cycle, and have received no further training. While the RL agents and the larger DT show good performance across all cycles, showing their ability to generalise, the smaller DT has generally a lower fuel economy and higher variance. The set of RL Agents is selected from Fig. 5. DT depths of 3 and 6 are chosen as a representation of increasing depth, and to avoid a crowded plot. The complete table of results for all depths is given in the supplementary material. *Below:* The faults associated with the above policies. Similar to the FE, the smaller DTs, in general, have a higher variance in the number of faults, indicating that the smaller depths risk decreased performance on new cycles that were not seen during training.

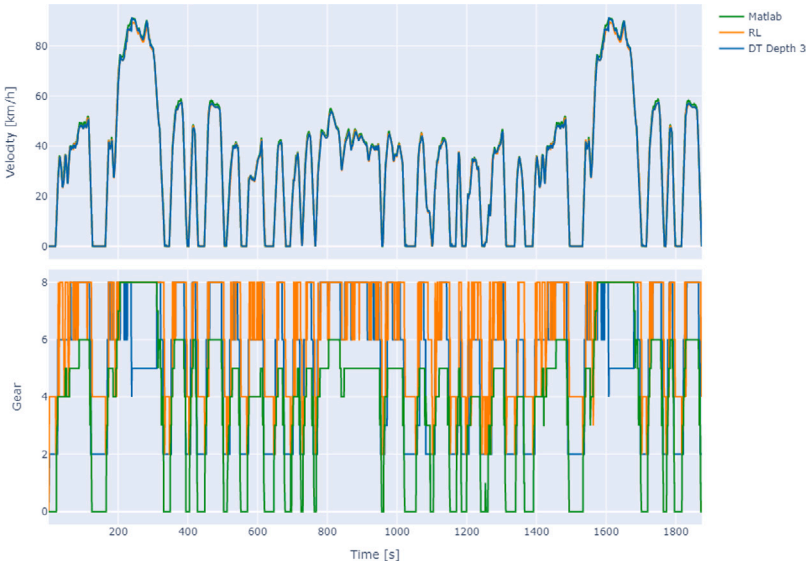

Fig. 8. *Above:* The velocity as a function of time for the FTP cycle. The reference velocity is given in green, with a trained RL policy shown in orange and the policy of a depth 3 DT in blue. The performance of these policies is shown in Table 3. *Below:* The associated shifting policies as a function of time. While the velocity differences between the policies are barely visible with this resolution, in the shifting maps of each policy we can clearly see that the RL Agent and the associated DT policy clearly stay in a higher gear at intermediate velocities. This is the source of the increase in FE. An artefact of the low DT depth is shown just before 300 s, where the DT shifts down to gear 5, where the benchmark and the RL Agent remain in gear 8. This can also be seen visually in the look-up table shown in.

Table 2
The set of driving cycles used in the evaluation of the different policies. The cycles are available online at United States Environmental Protection Agency (2023) and United Nations Economic Commission for Europe (2015).

Cycle name	Attributes
NYC	Low speed stop-and-go traffic conditions
HWFET	Highway driving conditions under 60 mph
J1015COL	Japanese exhaust emission and fuel economy driving schedule
US06	High acceleration aggressive driving schedule
ECE_P1	Economic Commission for Europe Part 1: Elementary Urban Cycle
ECE_P2	Economic Commission for Europe Part 2: Extra-Urban Driving Cycle
FTP	Federal Test Procedure, composed of the UDDS cycle
UDDS	City driving conditions
WLTC_class_1	Worldwide harmonised Light vehicles Test Procedure Low and medium speeds
WLTC_class_2	Worldwide harmonised Light vehicles Test Procedure Low, medium and high speeds
WLTC_class_3	Worldwide harmonised Light vehicles Test Procedure Low, medium, high and extra high speeds

Table 3
Example solution. Three agents are considered: a neural network based RL policy, the associated DT of depth 3, and the Matlab benchmark. The values of fuel economy (FE) and the number of faults are given when the agent is evaluated on the FTP cycle (used for training).

Agent	FE	Faults
RL	46.50	8
DT depth 3	40.88	0
Matlab	38.76	8

4.4. Single policy for FTP cycle

Having looked at the general behaviour of different RL and DT policies in the previous sections, here we consider one policy explicitly to compare the resulting shifting policies in more detail. The chosen policy is taken from the set of solutions shown in Fig. 6 and we choose the smallest DT (depth 3) which outperformed the original, MATLAB based benchmark on the FTP cycle. The evaluation of the chosen DT policy, in comparison to the associated RL policy and benchmark policy is given in Table 3.

The resulting RL neural network policy, the DT policy with depth 3, and the Matlab benchmark policy, are shown in Fig. 8. There, we compare the vehicle velocity of the three agents as well as look explicitly at the shifting policy of the agent. It is clear that the RL and DT policies are much more highly concentrated on higher gears, and shift gears much more frequently than the benchmark. This targeted up and down shifting allows for the increase in fuel economy without sacrificing the velocity following — the DT agent in this case has zero velocity faults. The high frequency shifting policy is a result that our simulation and reward function doesn't include any information about user comfort, something which most likely favours a lower shifting frequency. This feedback could be added indirectly (with a maximum number of shifts allowed in a given time window) or directly (by learning with direct human feedback), however this was not included for the purpose of this research.

To further derive explainability from these models, we can calculate the Shapley Additive Explanations (SHAP) for the resulting DT using the algorithm in Lundberg et al. (2020) and the InterpML package for Python (Nori et al., 2019). This value shows how each feature (velocity, engine speed and acceleration) contributes to the probability of being in each gear. The larger the absolute value of the SHAP value is, the more importance that feature has to be in that gear. For the DT policy used in Fig. 8 these values have been calculated and the resulting plots are given shown in Fig. 9. These values show us a clear relationship between the vehicle velocity, acceleration and the resulting gear. For higher velocities and lower acceleration there is a higher probability to be in a higher gear, while lower velocities or high acceleration gives a higher probability of being in a lower gear. This follows the intuition that in order to keep speed, downshifting is required to accelerate.

Table 4
The thresholds for the look-up table derived from the DT of depth 3 in Section 4.4.

Parameter	Velocity [km/h]	Engine speed [rpm]	Pedal position [0–1]
Thresholds	20.45	2091	0.5
	39.54	3485	
		3545	
		3667	

Table 5
The final look-up tables derived from the DT of depth 3. The thresholds for engine speed (ES), vehicle velocity (V) are given in each table individually, the threshold for the accelerator pedal position (PP) is the distinction between the upper and lower table.

PP $\leq$ 0.5	V $\leq$ 20.45	20.45 < V $\leq$ 39.54	39.54 < V
ES $\leq$ 2091	2	6	8
2091 < ES $\leq$ 3485	3	3	8
3485 < ES $\leq$ 3545	3	3	8
3545 < ES $\leq$ 3667	3	3	8
3667 < ES	3	3	5
0.5 < PP	V $\leq$ 20.45	20.45 < V $\leq$ 39.54	39.54 < V
ES $\leq$ 2091	2	6	6
2091 < ES $\leq$ 3485	3	3	6
3485 < ES $\leq$ 3545	3	3	4
3545 < ES $\leq$ 3667	3	3	4
3667 < ES	3	3	4

4.5. Software in the loop testing

In the above evaluation, the policies were implemented and evaluated in Python, using the precompiled shared library .dll file. In order to implement software in the loop testing, we implement the DT as a look-up table in the original simulation and evaluate it directly there. Note that while our RL agent was trained using 0.5 s (50 ms) updates, the original simulation uses the look-up table to update every 2 ms. This testing is done for the agents defined in Table 3. The DT of depth 3 can be converted to the look-up table with thresholds in Table 4.

The final look-up table, derived using the values from the final leaves of the DT, and the thresholds defined at each node, is shown in .

The results of implementing this table are given in Fig. 10. Interestingly, the implementation of the above look-up table brings to light one of the advantages that the interpretability of this approach has. In the above solution, there is one look-up table entry (corresponding to the input parameters PP $\leq$ 0.5, 39.54 < V, 3667 < ES) which gives the gear as 5. This can be seen directly in the shifting policy during the FTP cycle in Fig. 8. At time $t \approx 240$ the DT policy shifts down to gear 5, when both the RL Agent and the Matlab benchmark remain in gear 8. This downshifting at high speeds is counter intuitive to human experience, and is noticeable when looking at the look-up table directly. Because our approach uses RL in the creation of the look-up table, there is still room for control engineers to visually inspect the resulting table, and manually adjust entries when needed. In this case, this value could be changed to gear 8 and the engineers could verify that this improves the resulting policy.

The policy shown in has a total of 30 entries, which is a direct result of choosing a DT of depth 3. While there is no clearly defined limit at which size a look-up table becomes too large to interpret, the smaller the table is, the easier it is to implement technically and to be verified by test engineers. If a DT of higher depth was taken in order to improve performance, the look-up table size would necessarily increase. A comparison and discussion of the DT depth and number of look-up table entries for this example is given in ??.

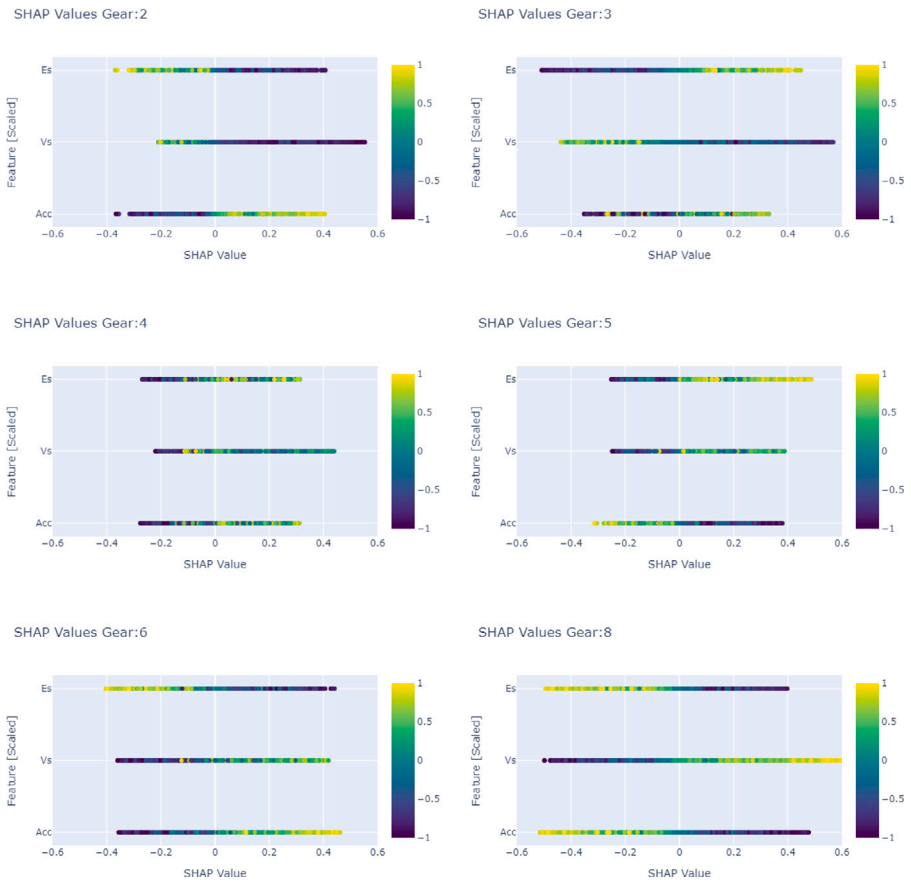

Fig. 9. SHAP values for each gear in the policy of the DT shown in Section 4.4 for the state space variables: Acceleration (Acc), Velocity (Vs) and Engine Speed (Es). As expected intuitively, the higher the velocity and the lower the acceleration, the higher the probability to be in a higher gear. Conversely, a low velocity or a high acceleration increases the probability to be in a lower gear. The crossover between these two regimes happen around gear 4.

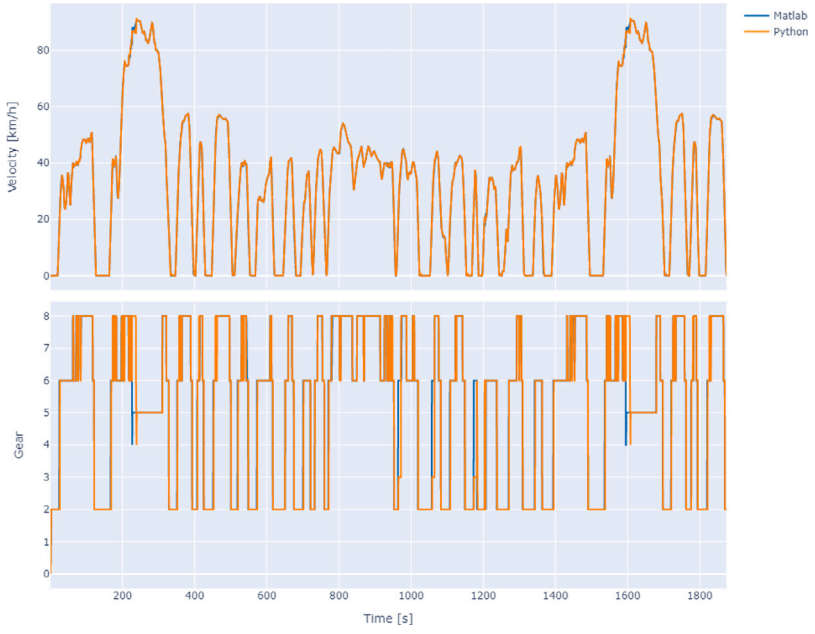

Fig. 10. *Above:* The velocity as a function of time for the FTP cycle in blue (the look-up table reimplemented in simulink), in orange (the look-up table in the Python compiled simulation). *Below:* The associated shifting policies as a function of time.

5. Conclusions

In this paper we investigated a post-hoc explainable AI approach, based on a derivation of DTs and ultimately, look-up tables from an original neural network. This directly allows us to exploit the benefits of RL in a way where it remains feasible to implement the resulting policy in a practical, control engineering use case. The proposed method consists of initially training an RL agent using the PPO algorithm, then sampling the resulting policy and training a DT, which has been limited to a manageable depth. This DT is then translated into a look-up table, which can be directly implemented in control software. This method was demonstrated in a powertrain control use case, in the gear shifting logic of an automatic transmission. In order to study the performance of the DT policies as a function of depth, we use a set of different optimised RL policies, and trained sets of DTs with depths varying from 1–20. We see the stable convergence of the DT behaviour for a unique solution of the RL Agent around depth 9, with single solutions of significantly smaller depths (depth 3) already outperforming the benchmark policy in both fuel economy and velocity following. Looking at one such policy in more depth we can see the differences directly in the policy which allow for improved performance. Implementing the associated look-up table (with 7 thresholds over the three input variables) we tested this new policy in a software in the loop setup. Each step takes advantage of standard Python libraries thus reducing the required engineering resources, and allowing for standardised algorithms to be used.

Our results here are based on a reward that includes a trade-off between fuel economy and the number of velocity faults. What determines this trade-off is dependent on the end-users (Ahmad et al., 2018) which is, actually, already a part of automotive design. More specifically, this trade-off is part of the design of different operating modes, with most engines offering an ‘eco’ or a ‘sport’ mode which allows the end consumer to decide this trade-off themselves. While the reward used here features two easily quantifiable aspects – fuel economy and velocity faults – factors that impact the driver’s comfort, such as impacting driver comfort, such as the frequency of gear changes and the wear-and-tear from excessive shifting, are more challenging to measure objectively and were therefore excluded from the RL agent’s objectives. Addressing these aspects would require input from control engineers, and the policy would need adjustment to account for them. Future work could explore incorporating these factors into the reward function, for instance through user feedback or by setting constraints on the agent, such as limiting the frequency of gear shifts.

In addition, our results here are based on the fact that, due to the small dimension of the input state space, a uniform sampling of the RL policy was possible. Small state spaces are common in industrial applications, as they are currently being implemented with simpler control solutions. However, in the case of higher dimensional state spaces this method is not scalable and a more in depth study of this pipeline is required, and will be the subject of future work. In this case, sampling can be done via a variety of different methods, e.g., those outlined in Bastani et al. (2018) and Ross et al. (2011).

Finally, a set of different driving cycles have been used to independently test how well the policies generalise to unseen circumstances, having been trained on a different cycle. While these driving cycles have been specifically designed to reflect real-world driving scenarios, a complete performance evaluation requires tests in real-world scenarios outside of the simulation. Such tests will be the subject of future work.

While these results are given for one use case, the method can be generalised to different use cases, providing low-barrier access to benefits from RL in cases where implementing a neural network is unrealistic. This is particularly relevant to safety-critical use cases such as medical control use cases where a neural network based policy is not able to satisfy the regulatory requirements. Additionally, even in non safety-critical industrial use cases, where decisions made in process control should be traceable, this approach could allow RL to be implemented without sacrificing the accountability from industrial process optimisation to the parameter configuration of a consortium blockchain (see Zhai et al., 2024).

CRediT authorship contribution statement

C. Laflamme: Writing – original draft, Visualization, Validation, Software, Project administration, Funding acquisition, Formal analysis, Data curation, Conceptualization. **J. Doppler:** Writing – review & editing, Visualization, Validation, Software, Resources, Formal analysis, Data curation. **B. Palvolgyi:** Software. **S. Dominka:** Writing – review & editing, Validation, Supervision, Conceptualization. **Zs.J. Viharos:** Writing – review & editing, Validation, Supervision, Conceptualization. **S. Haeussler:** Writing – review & editing, Validation, Supervision, Conceptualization.

Funding

The research leading to these results has received funding from the ICT of the Future programme (project number 887500). ICT of the Future is a research, technology and innovation funding programme of the Republic of Austria, Ministry of Climate Action. The Austrian Research Promotion Agency (FFG) has been authorised for the programme management. The research was supported also by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory and party by the European Commission, through the H2020 project EPIC (https://www.centre-epic.eu/) under grant No. 739592 and by the ED- 18-2-2018-0006 grant “Research on prime exploitation of the potential provided by the industrial digitalisation”.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Zs. J. Viahors reports financial support was provided by Artificial Intelligence National Laboratory. Zs. J. Viharos reports financial support was provided by National Research Development and Innovation Office. Zs. J. Viharos reports financial support was provided by H2020 Teaming of excellent research institutions and low performing RDI regions. C. Laflamme reports financial support was provided by Austrian Research Promotion Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2025.110135.

Data availability

Data will be made available on request.

References

Ahmad, M.A., Eckert, C., Teredesai, A., 2018. Interpretable machine learning in healthcare. In: Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics. pp. 559–560.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, Francisco Herrera, 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion 58, 82–115. http://dx.doi.org/10.1016/j.inffus.2019.12.012.

Andrews, R., Diederich, J., Tickle, A.B., 1995. Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl.-Based Syst. 8 (6), 373–389.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, Noah Dormann, 2021. Stable-Baselines3: Reliable reinforcement learning implementations. J. Mach. Learn. Res. 22 (268), 1–8, URL: http://jmlr.org/papers/v22/20-1364.html.

Aradi, S., 2022. Survey of deep reinforcement learning for motion planning of autonomous vehicles. IEEE Trans. Intell. Transp. Syst. 23 (2), 740–759. http://dx.doi.org/10.1109/TITS.2020.3024655.

Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al., 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion 58, 82–115.

Ayoub, J., Yang, X.J., Zhou, F., 2021. Modeling dispositional and initial learned trust in automated vehicles with predictability and explainability. Transp. Res. F 77, 102–116.

Badr Ben Elallid, Nabil Benamar, Abdelhakim Senhaji Hafid, Tajjeeddine Rachidi, Nabil Mrani, 2022. A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving. J. King Saud Univ. - Comput. Inf. Sci. 34 (9), 7366–7390. http://dx.doi.org/10.1016/j.jksuci.2022.03.013.

Bastani, O., Pu, Y., Solar-Lezama, A., 2018. Verifiable reinforcement learning via policy extraction. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. vol. 31, Curran Associates, Inc.

Bautista-Montesano, R., Bustamante-Bello, R., Ramirez-Mendoza, R.A., 2020. Explainable navigation system using fuzzy reinforcement learning. Int. J. Interact. Des. Manuf. (IJIDeM) 14 (4), 1411–1428.

Biewald, L., 2020. Experiment tracking with weights and biases. URL: https://www.wandb.com/.

Breiman, L., Friedman, J., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees, first ed. Chapman and Hall/CRC, http://dx.doi.org/10.1201/9781315139470.

Chen, I.-M., Zhao, C., Chan, C.-Y., 2019. A deep reinforcement learning-based approach to intelligent powertrain control for automated vehicles. In: 2019 IEEE Intelligent Transportation Systems Conference. ITSC, pp. 2620–2625. http://dx.doi.org/10.1109/ITSC.2019.8917076.

Coppens, Y., Efthymiadis, K., Lenaerts, T., Nowé, A., Miller, T., Weber, R., Magazzeni, D., 2019. Distilling deep reinforcement learning policies in soft decision trees. In: Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence. pp. 1–6.

Dai, Y., Chen, Q., Zhang, J., Wang, X., Chen, Y., Gao, T., Xu, P., Chen, S., Liao, S., Jiang, H., et al., 2022. Enhanced oblique decision tree enabled policy extraction for deep reinforcement learning in power system emergency control. Electr. Power Syst. Res. 209, 107932.

Dominka, S., Doppler, J., Laflamme, C., Litschauer, T., 2024. Learning to drive (efficiently). In: Accepted At the 2024 IEEE International Conference on Electro/Information Technology.

Ericsson, E., 2001. Independent driving pattern factors and their influence on fuel-use and exhaust emission factors. Transp. Res. D 6 (5), 325–345.

Estrada, P.M., de Lima, D., Bauer, P.H., Mammetti, M., Bruno, J.C., 2023. Deep learning in the development of energy management strategies of hybrid electric vehicles: A hybrid modeling approach. Appl. Energy 329, 120231.

Fontaras, G., Zacharof, N.-G., Ciuffo, B., 2017. Fuel consumption and CO2 emissions from passenger cars in Europe–laboratory versus real-world emissions. Prog. Energy Combust. Sci. 60, 97–131.

Frosst, N., Hinton, G., 2017. Distilling a neural network into a soft decision tree. arXiv preprint arXiv:1711.09784.

Garg, P., Silvas, E., Willems, F., 2023. Systematic hyperparameter selection in machine learning-based engine control to minimize calibration effort. Control Eng. Pract. 140, 105666.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D., 2018. A survey of methods for explaining black box models. ACM Comput. Surv. 51 (5), 1–42.

He, H., Meng, X., Wang, Y., Khajepour, A., An, X., Wang, R., Sun, F., 2024. Deep reinforcement learning based energy management strategies for electrified vehicles: Recent advances and perspectives. Renew. Sustain. Energy Rev. 192, 114248.

Henriksson, J., Borg, M., Englund, C., 2018. Automotive safety and machine learning: Initial results from a study on how to adapt the ISO 26262 safety standard. In: Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems. pp. 47–49.

Hickling, T., Zenati, A., Aouf, N., Spencer, P., 2023. Explainability in deep reinforcement learning: A review into current methods and applications. ACM Comput. Surv. 56 (5), 1–35.

Hu, B., Li, J., 2021. An edge computing framework for powertrain control system optimization of intelligent and connected vehicles based on curiosity-driven deep reinforcement learning. IEEE Trans. Ind. Electron. 68 (8), 7652–7661. http://dx.doi.org/10.1109/TIE.2020.3007100.

Hu, D., Xie, H., Song, K., Zhang, Y., Yan, L., 2023. An apprenticeship-reinforcement learning scheme based on expert demonstrations for energy management strategy of hybrid electric vehicles. Appl. Energy 342, 121227.

International Organization for Standardization, 2018. Road vehicles – functional safety – part 6: Product development at the software level.

Jia, S., Lin, P., Li, Z., Zhang, J., Liu, S., 2020a. Visualizing surrogate decision trees of convolutional neural networks. J. Vis. 23 (1), 141–156. http://dx.doi.org/10.1007/s12650-019-00607-z.

Jia, X., Peng, J., Liu, Y., Liu, B., Wang, P., Lu, Y., Wen, M., Huang, Z., 2020b. Car-following safe headway strategy with battery-health conscious: A reinforcement learning approach. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics. SMC, IEEE, pp. 2432–2437.

Kerbel, L., Ayalew, B., Ivanco, A., 2023. Adaptive policy learning for data-driven powertrain control with eco-driving. Eng. Appl. Artif. Intell. 124, 106489.

Khalatbarisoltani, A., Kandidayeni, M., Boulon, L., Hu, X., 2022. A decentralized multi-agent energy management strategy based on a look-ahead reinforcement learning approach. SAE Int. J. Electrified Veh. 11 (2).

Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P., 2022. Deep reinforcement learning for autonomous driving: A survey. IEEE Trans. Intell. Transp. Syst. 23 (6), 4909–4926. http://dx.doi.org/10.1109/TITS.2021.3054625.

Koch, L., Picerno, M., Badalian, K., Lee, S.-Y., Andert, J., 2023. Automated function development for emission control with deep reinforcement learning. Eng. Appl. Artif. Intell. 117, 105477.

Lal, G.R., Mithal, V., 2022. NN2rules: Extracting rule list from neural networks.

Li, Y., He, H., Khajepour, A., Wang, H., Peng, J., 2019a. Energy management for a power-split hybrid electric bus via deep reinforcement learning with terrain information. Appl. Energy 255, 113762.

Li, Y., He, H., Peng, J., Wang, H., 2019b. Deep reinforcement learning-based energy management for a series hybrid electric vehicle enabled by history cumulative trip information. IEEE Trans. Veh. Technol. 68 (8), 7416–7430.

Liang, Z., Ruan, J., Wang, Z., Liu, K., Li, B., 2024. Soft actor-critic-based EMS design for dual motor battery electric bus. Energy 288, 129849.

Lin, Y., Chu, L., Hu, J., Hou, Z., Li, J., Jiang, J., Zhang, Y., 2023. Progress and summary of reinforcement learning on energy management of MPS-EV. Heliyon.

Lindsey Kerbel, Beshah Ayalew, Andrej Ivanco, 2023. Adaptive policy learning for data-driven powertrain control with eco-driving. Eng. Appl. Artif. Intell. 124, 106489. http://dx.doi.org/10.1016/j.engappai.2023.106489.

Louback, E., Biswas, A., Machado, F., Emadi, A., 2024. A review of the design process of energy management systems for dual-motor battery electric vehicles. Renew. Sustain. Energy Rev. 193, 114293.

Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.-I., 2020. From local explanations to global understanding with explainable AI for trees. Nat. Mach. Intell. 2 (1), 56–67.

Lundberg, S.M., Nair, B., Vavilala, M.S., Horibe, M., Eisses, M.J., Adams, T., Liston, D.E., Low, D.K.-W., Newman, S.-F., Kim, J., et al., 2018. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. Nat. Biomed. Eng. 2 (10), 749–760.

Milani, S., Topin, N., Veloso, M., Fang, F., 2024. Explainable reinforcement learning: A survey and comparative review. ACM Comput. Surv. 56 (7), http://dx.doi.org/10.1145/3616864.

Müller, P., Faber, L., Martinkus, K., Wattenhofer, R., 2022. DT+GNN: A fully explainable graph neural network using decision trees.

Munkhdalai, L., Wang, L., Park, H.W., Ryu, K.H., 2019. Advanced neural network approach, its explanation with lime for credit scoring application. In: Asian Conference on Intelligent Information and Database Systems. Springer, pp. 407–419.

Nori, H., Jenkins, S., Koch, P., Caruana, R., 2019. Interpretml: A unified framework for machine learning interpretability. arXiv preprint arXiv:1909.09223.

Norouzi, A., Heidarifar, H., Borhan, H., Shahbakhti, M., Koch, C.R., 2023. Integrating machine learning and model predictive control for automotive applications: A review and future directions. Eng. Appl. Artif. Intell. 120, 105878.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Qi, C., Zhu, Y., Song, C., Cao, J., Xiao, F., Zhang, X., Xu, Z., Song, S., 2021. Self-supervised reinforcement learning-based energy management for a hybrid electric vehicle. J. Power Sources 514, 230584.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016. " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144.

Ross, S., Gordon, G., Bagnell, D., 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 627–635.

Roth, A.M., Topin, N., Jamshidi, P., Veloso, M., 2019. Conservative Q-improvement: Reinforcement learning for an interpretable decision-tree policy.

Silva, A., Gombolay, M., Killian, T., Jimenez, I., Son, S.-H., 2020. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In: Chiappa, S., Calandra, R. (Eds.), Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. In: Proceedings of Machine Learning Research, vol. 108, PMLR, pp. 1855–1865.

Silva, A., Killian, T., Rodriguez, I.D.J., Son, S.-H., Gombolay, M., 2019. Optimization methods for interpretable differentiable decision trees in reinforcement learning. arXiv preprint arXiv:1903.09338.

Simulink, 2024a. URL: https://www.mathworks.com/help/autoblks/ug/explore-the-conventional-vehicle-reference-application.html.

Simulink, 2024b. URL: https://www.mathworks.com/help/autoblks/ug/optimize-transmission-control-module-shift-schedules.html.

Tabani, H., Kosmidis, L., Abella, J., Cazorla, F.J., Bernat, G., 2019. Assessing the adherence of an industrial autonomous driving framework to iso 26262 software guidelines. In: Proceedings of the 56th Annual Design Automation Conference 2019. pp. 1–6.

Tao, F., Gong, H., Fu, Z., Guo, Z., Chen, Q., Song, S., 2023. Terrain information-involved power allocation optimization for fuel cell/battery/ultracapacitor hybrid electric vehicles via an improved deep reinforcement learning. Eng. Appl. Artif. Intell. 125, 106685.

The European Commission, 2017. Commission regulation (EU) 2017/1151 of 1 june 2017. URL: https://eur-lex.europa.eu/eli/reg/2017/1151/oj.

Tian, X., Tao, F., Fu, Z., Zhu, L., Sun, H., Song, S., 2024. Optimizing fuel economy of fuel cell hybrid electric vehicle based on energy management strategy with integrated rapid thermal regulation. Eng. Appl. Artif. Intell. 132, 107880.

Tóth, S.H., Viharos, Z.J., Bárdos, Á., Szalay, Z., 2024. Sim-to-real application of reinforcement learning agents for autonomous, real vehicle drifting. Veh. 6 (2), 781–798. http://dx.doi.org/10.3390/vehicles6020037, URL: https://www.mdpi.com/2624-8921/6/2/37.

Towell, G.G., Shavlik, J.W., 1993. Extracting refined rules from knowledge-based neural networks. Mach. Learn. 13, 71–101.

Towers, M., Terry, J.K., Kwiatkowski, A., Balis, J.U., de Cola, G., Deleu, T., Goulão, M., Kallinteris, A., KG Arjun, Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Shen, A.T.J., Younis, O.G., 2023. Gymnasium. http://dx.doi.org/10.5281/zenodo.8127026, URL: https://zenodo.org/record/8127025.

United Nations Economic Commission for Europe, 2015. Worldwide harmonized light vehicles test procedure. URL: https://unece.org/fileadmin/DAM/trans/doc/2012/wp29grpe/WLTP-DHC-12-07e.xls.

United States Environmental Protection Agency, 2023. Dynamometer drive schedules. URL: https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules.

Verma, A., Murali, V., Singh, R., Kohli, P., Chaudhuri, S., 2018. Programmatically interpretable reinforcement learning. In: International Conference on Machine Learning. pp. 5045–5054.

Viharos, Z.J., Jakab, R., 2021. Reinforcement learning for statistical process control in manufacturing. Meas. 182, 109616. http://dx.doi.org/10.1016/j.measurement.2021.109616.

Viharos, Z.J., Kis, K.B., 2015. Survey on neuro-fuzzy systems and their applications in technical diagnostics and measurement. Meas. 67, 126–136.

Vinícius G. Costa, Jorge Pérez-Aracil, Sancho Salcedo-Sanz, Carlos E. Pedreira, 2024. Evolving interpretable decision trees for reinforcement learning. Artificial Intelligence 327, 104057. http://dx.doi.org/10.1016/j.artint.2023.104057.

Vouros, G.A., 2022. Explainable deep reinforcement learning: state of the art and challenges. ACM Comput. Surv. 55 (5), 1–39.

Wan, A., Dunlap, L., Ho, D., Yin, J., Lee, S., Petryk, S., Bargal, S.A., Gonzalez, J.E., 2020. NBDT: Neural-backed decision tree. In: International Conference on Learning Representations.

Wang, Y., Wu, Y., Tang, Y., Li, Q., He, H., 2023. Cooperative energy management and eco-driving of plug-in hybrid electric vehicle via multi-agent reinforcement learning. Appl. Energy 332, 120563.

Yang, B.-B., Shen, S.-Q., Gao, W., 2019. Weighted oblique decision trees. Proc. AAAI Conf. Artif. Intell. 33 (01), 5621–5627. http://dx.doi.org/10.1609/aaai.v33i01.33015621.

Ye, Y., Wang, H., Xu, B., Zhang, J., 2023. An imitation learning-based energy management strategy for electric vehicles considering battery aging. Energy 283, 128537.

Zhai, Z., Shen, S., Mao, Y., 2024. An explainable deep reinforcement learning algorithm for the parameter configuration and adjustment in the consortium blockchain. Eng. Appl. Artif. Intell. 129, 107606.

Zhang, H., Fan, Q., Liu, S., Li, S.E., Huang, J., Wang, Z., 2021. Hierarchical energy management strategy for plug-in hybrid electric powertrain integrated with dual-mode combustion engine. Appl. Energy 304, 117869.

Zhang, K., Ruan, J., Li, T., Cui, H., Wu, C., 2023. The effects investigation of data-driven fitting cycle and deep deterministic policy gradient algorithm on energy management strategy of dual-motor electric bus. Energy 269, 126760.

Zhang, H., Zhou, A., Lin, X., 2020. Interpretable policy derivation for reinforcement learning based on evolutionary feature synthesis. Complex Intell. Syst. 6, 741–753.

Zhou, F., Alsaid, A., Blommer, M., Curry, R., Swaminathan, R., Kochhar, D., Talamonti, W., Tijerina, L., 2022a. Predicting driver fatigue in monotonous automated driving with explanation using GPBoost and SHAP. Int. J. Hum.– Comput. Interact. 38 (8), 719–729.

Zhou, J., Xue, Y., Xu, D., Li, C., Zhao, W., 2022b. Self-learning energy management strategy for hybrid electric vehicle via curiosity-inspired asynchronous deep reinforcement learning. Energy 242, 122548.

Zhou, J., Xue, S., Xue, Y., Liao, Y., Liu, J., Zhao, W., 2021. A novel energy management strategy of hybrid electric vehicle via an improved TD3 deep reinforcement learning. Energy 224, 120118.