Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Development of a Neural Network for Optimal Train Power Control Using Reinforcement Learning

Master’s Final Degree Project

Justas Janickas

Project author

Assoc. Prof. Dr. Paulius Palevičius

Supervisor

Kaunas, 2026

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Development of a Neural Network for Optimal Train Power Control Using Reinforcement Learning

Master's Final Degree Project

Data Science and Artificial Intelligence (6211AX013)

Justas Janickas

Project author

Assoc. Prof. Dr. Paulius Palevičius

Supervisor

Assoc. Prof. Dr. Karolina Armonaitė

Reviewer

Kaunas, 2026

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Justas Janickas

Development of a Neural Network for Optimal Train Power Control Using Reinforcement Learning

Declaration of Academic Integrity

Justas Janickas. Development of a Neural Network for Optimal Train Power Control Using Reinforcement Learning. Master's Final Degree Project / supervisor Assoc. Prof. Dr. Paulius Palevičius; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Data Science and Artificial Intelligence (Mathematical Sciences).

Keywords: train control; energy consumption; reinforcement learning; route segmentation; slope lookahead.

Kaunas, 2026. 65 p.

Summary

This study presents a reinforcement learning approach for reducing energy consumption in freight train operations while still meeting scheduled arrival times. A simplified physical simulation, combined with a learned power control policy, allows the system to select engine power levels based on the remaining travel time, speed limits, train mass, and the upcoming terrain along the route. The model is trained through repeated interaction with the simulated environment, where it learns to balance punctuality and energy consumption based on the reward function designed for this task.

The results show that the trained model can learn energy-efficient driving behaviour and outperform basic control strategies in terms of energy consumption. Despite relying on simplified models and route-specific training, the work demonstrates that reinforcement learning is a practical and effective tool for supporting train operators in executing energy-efficient train power control.

Justas Janickas. Skatinamuoju mokymusi grįsto neuroninio tinklo kūrimas optimaliam traukinio galios valdymui. Magistro baigiamasis projektas / vadovas doc. dr. Paulius Palevičius; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Duomenų mokslas ir dirbtinis intelektas (Matematikos mokslai).

Reikšminiai žodžiai: traukinio valdymas; kuro sąnaudos; skatinamasis mokymasis; maršruto sudalijimas; nuolydžio numatymas.

Kaunas, 2026. 65 p.

Santrauka

Šiame darbe pristatomas skatinamojo mokymosi metodas, skirtas sumažinti krovininių traukinių energijos sąnaudas, išlaikant judėjimo grafikų keliamus reikalavimus. Supaprastinta fizinė simuliacija kartu su išmokta galios valdymo strategija leidžia sistemai parinkti variklio galios lygius pagal likusį kelionės laiką, greičio ribojimus, traukinio masę ir numatomą maršruto reljefą. Modelis apmokomas simuliuotant aplinką, kurioje jis išmoksta derinti punktualumą ir energijos sąnaudas, naudodamas šiai užduočiai suformuotą atlygio funkciją.

Gautuose rezultatuose matome, kad modelis gali išmokti efektyvaus važiavimo dėsningumų ir pagal kuro sąnaudas pranokti paprastas galios parinkimo strategijas. Nors darbas remiasi supaprastintais modeliais ir vykdo mokymąsi vienam konkrečiam maršrutui, jis parodo, kad skatinamasis mokymasis yra praktiška ir veiksminga priemonė, galinti traukinių vairuotojams parekomenduoti traukinio galios valdymo parinkimą, kuris sumažintų kuro sąnaudas.

Contents

List of Figures

Introduction

Although having a long history, rail transport to this day remains an important component of the global freight system, capable of carrying high cargo capacity while achieving lower energy consumptions compared to road transport. Despite its current efficiency, there is still significant potential to reduce fuel consumption and emissions through improved operational strategies.

Quite commonly, freight train control is performed by an operator person who is responsible for lowering the energy consumption through their experience alone. Although some of the natural choices, such as coasting in downhill sections, can save fuel, determining the optimal power for each segment depends on multiple factors, including train mass, track gradient, speed limits, and time target for the trip, further increasing the challenge of selecting an optimal control mode.

Instead of formulating a mathematical optimisation problem, recent developments in artificial intelligence, particularly reinforcement learning, provide an alternative approach to solving the challenge. Reinforcement learning enables systems to improve control strategies through interaction with the environment, considering both immediate and long-term benefits, this way providing recommendations to the operator on how to adjust engine power in order to reduce energy consumption.

The aim of this study is to investigate how the application of reinforcement learning can be used to optimise energy consumption in freight trains while complying with the time targets set by the schedule.

The tasks of this work are as follows.

1. To implement a simplified physical model for train movement simulation based on the law of conservation of energy;
2. To build a neural network model based on the reinforcement learning methodology for engine power level selection at different positions along the route;
3. To design a method for encoding lookahead information of the height profile paired with other relevant metadata about the upcoming route segments;
4. To construct a reward function that captures good practices of punctual and energy-efficient operation;
5. To generalise the model to account for different masses of the train, thus eliminating the need to retrain the model in case of small adjustments to cargo load.
6. To propose a method for dividing the spatial information of the route into segments in which a constant driving mode should be maintained;
7. To evaluate the fuel consumption improvements achieved by the reinforcement learning solution against a selection of simple operational strategies.

1 Literature Review

Throughout the last century of research, our understanding of train motion physics and the aim for energy efficiency have progressed from purely mathematical and analytical calculations to practical applications in industry. Transport companies aim to reduce operational costs and negative environmental impact by designing more and more efficient systems. However, there is great potential for improvement in the application of smart driving strategies that can make full use of modern systems.

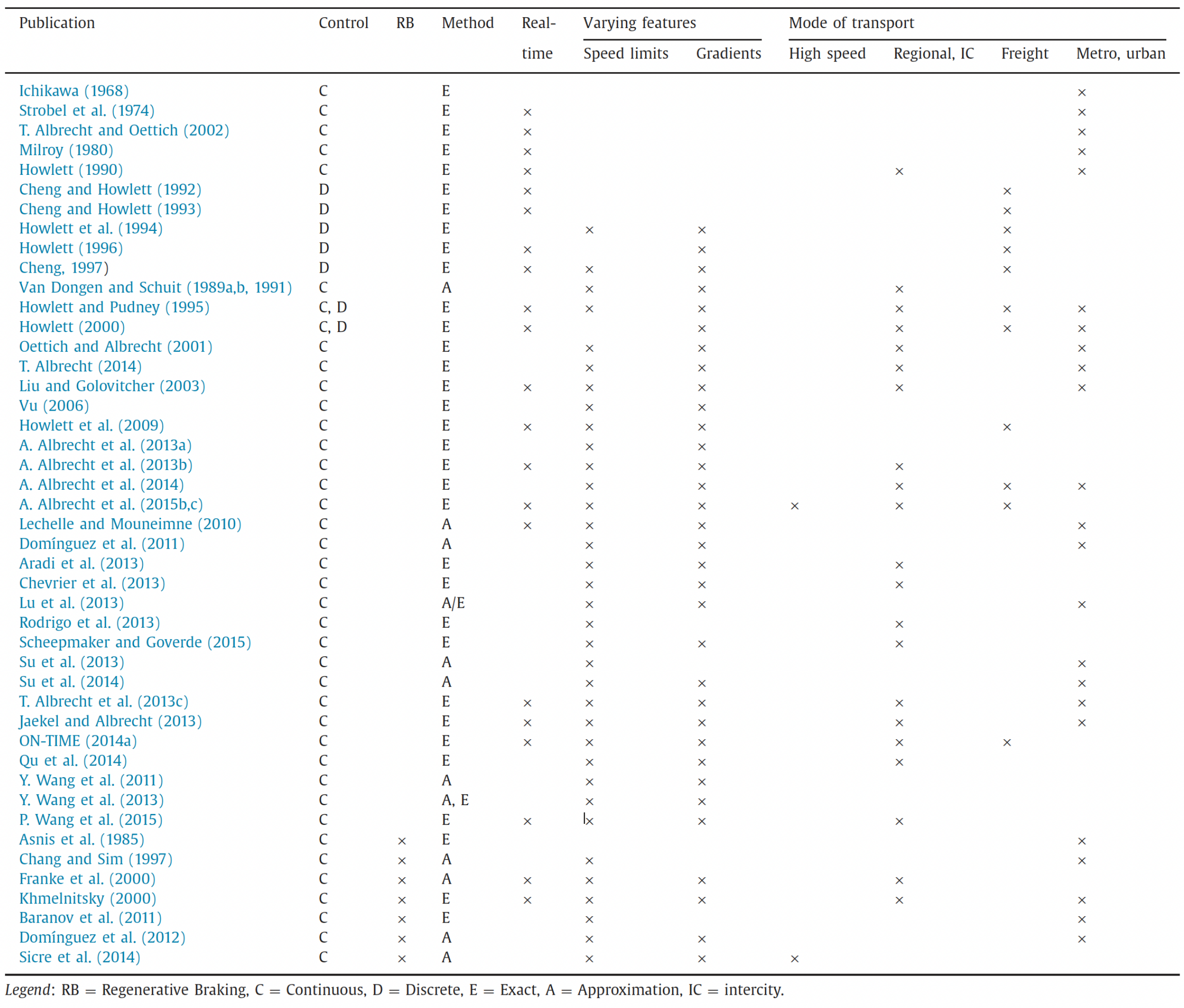
Reinforcement learning has established itself as a flexible method for solving various real-world problems that have complicated set-ups and clear end goals. By interacting with a simulated environment, a system can learn optimal actions without solving complex equations, thus acting as a recommendation system for close-to-optimal train operation. Familiarising ourselves with previous studies in the field enables us to avoid unnecessary duplication of work and build on top of it to meaningfully expand the public knowledge about the energy-efficient train control. In addition, existing studies often serve as a source of inspiration when designing novel solutions to the problem at hand.

In this chapter, we provide an overview of train motion optimisation improvements and reinforcement learning improvements throughout the literature. In Section 1.1 we present the background for energy-efficient train control and how the most influential works shaped the understanding of the optimal driving modes in rail transport. Section 1.2 then compares two different approaches to physical modelling used in the literature and the compromises they encompass. We then present a systematic grouping of different optimisation algorithms in Section 1.3 as well as the works that apply these algorithms in solving the energy-efficient train control problem. In later sections, we transition to the field of reinforcement learning. Section 1.4 presents the foundation work in the field of reinforcement learning itself, while Section 1.5 summarises the areas in transportation and, more specifically, in rail transport where reinforcement learning has been applied. Later in Section 1.6 we perform a comprehensive comparison of two recent articles applying reinforcement learning for energy-efficient train control. Finally, we review some of the practical aspects of reinforcement learning applications, namely, route segmentation and heigh profile lookahead encoding in Section 1.7 and Section 1.8, respectively.

1.1 Optimal Control and Switching Cases

Throughout the different studies on optimal train control, a variety of assumptions and operational constraints were considered. In order to clearly summarise the advances in this field, Scheepmaker et al.[1] present an extensive review of the literature up to 2017 on energy-efficient train control and the related topic of energy-efficient timetabling. They comment that most of the research is based on Pontryagin's Maximum Principle[2] formulated in 1962 and uses a fixed number of driving modes, although the selection of modes differs from paper to paper.

Since the first application of Pontryagin's Maximum Principle to train operations in 1968 by Ichikawa[3], where they considered simple conditions such as constant elevation and fixed speed limits, researchers continued to expand the formulation of the problem with more and more realistic constraints. The table in Figure 1 by Scheepmaker et al.[1] systemically groups the works into categories based on different aspects of problem formulation, as explained in the legend.



Publication	Control	RB	Method	Real-time	Varying features: Speed limits	Varying features: Gradients	Mode of transport: High speed	Mode of transport: Regional, IC	Mode of transport: Freight	Mode of transport: Metro, urban
Ichikawa (1968)	C		E							×
Strobel et al. (1974)	C		E	×						×
T. Albrecht and Oettich (2002)	C		E	×						×
Milroy (1980)	C		E	×						×
Howlett (1990)	C		E	×				×		×
Cheng and Howlett (1992)	D		E	×					×	
Cheng and Howlett (1993)	D		E	×					×	
Howlett et al. (1994)	D		E		×	×			×	
Howlett (1996)	D		E	×		×			×	
Cheng, 1997)	D		E	×	×	×			×	
Van Dongen and Schuit (1989a,b, 1991)	C		A		×	×		×		
Howlett and Pudney (1995)	C, D		E	×	×	×		×	×	×
Howlett (2000)	C, D		E	×		×		×	×	×
Oettich and Albrecht (2001)	C		E		×	×		×		×
T. Albrecht (2014)	C		E		×	×		×		×
Liu and Golovitcher (2003)	C		E	×	×	×		×		×
Vu (2006)	C		E		×	×				
Howlett et al. (2009)	C		E	×	×	×			×	
A. Albrecht et al. (2013a)	C		E		×	×				
A. Albrecht et al. (2013b)	C		E	×	×	×		×		
A. Albrecht et al. (2014)	C		E		×	×		×	×	×
A. Albrecht et al. (2015b,c)	C		E	×	×	×	×	×	×	
Lechelle and Mouneimne (2010)	C		A	×	×	×				×
Domínguez et al. (2011)	C		A		×	×				×
Aradi et al. (2013)	C		E		×	×		×		
Chevrier et al. (2013)	C		E		×	×		×		
Lu et al. (2013)	C		A/E		×	×				×
Rodrigo et al. (2013)	C		E		×			×		
Scheepmaker and Goverde (2015)	C		E		×	×		×		
Su et al. (2013)	C		A		×					×
Su et al. (2014)	C		A		×	×				×
T. Albrecht et al. (2013c)	C		E	×	×	×		×		×
Jaekel and Albrecht (2013)	C		E	×	×	×		×		×
ON-TIME (2014a)	C		E	×	×	×		×	×	
Qu et al. (2014)	C		E		×	×		×		
Y. Wang et al. (2011)	C		A		×	×				
Y. Wang et al. (2013)	C		A, E		×	×				
P. Wang et al. (2015)	C		E	×	×	×		×		
Asnis et al. (1985)	C	×	E							×
Chang and Sim (1997)	C	×	A		×					×
Franke et al. (2000)	C	×	A	×	×	×		×		
Khmelnitsky (2000)	C	×	E	×	×	×		×		×
Baranov et al. (2011)	C	×	E		×					×
Domínguez et al. (2012)	C	×	A		×	×				×
Sicre et al. (2014)	C	×	A		×	×	×			

Legend: RB = Regenerative Braking, C = Continuous, D = Discrete, E = Exact, A = Approximation, IC = intercity.

Fig. 1. Literature review summary by Scheepmaker et al.[1]

Later in 2020, Scheepmaker and Goverde[4] extend the classical optimal-control framework by explicitly considering the practical aspects of regenerative braking in energy-efficient train control. They model the optimal sequence of driving modes taking into account that regenerative braking has limited efficiency and depends on the train speed. Their results show that including realistic regenerative braking alters the optimal speed profile, leading to lower cruising speeds, shorter coasting phases, and earlier braking. They also demonstrate how these practical considerations can significantly influence energy consumption compared to strategies that assume ideal or purely mechanical braking.

More recently, Heineken et al.[5] revisits the classical optimal-control framework via Pontryagin's Maximum Principle, expands on it, and refines the switching schemes between driving modes for modern trains with regenerative braking. They base their algorithm on the work of Khmelnitsky[6] and provide a complete list of feasible switching cases between driving modes while using numerical examples and a comparison with data from 100 real-life passenger trains. Their work thus demonstrate that the optimised control strategy can reduce energy demand substantially compared to data collected from unassisted driving.

1.2 Physical Models

In order to accurately simulate train motion, a number of assumptions have to be made about the physics governing it. Even though trains usually consist of a series of linked waggons, the majority of research choose to abstract entirety of the train into a single point mass to make simulations more feasible.

Nold et al.[7] discuss that when simplified models are considered, they usually conform to the four driving modes of optimal energy-efficient train control. However, they raise the claim that researchers rarely expand the problem to account for more complex models and their traction components. When such more complex conditions are considered, the derived optimal trajectories tend not to perfectly follow the four optimal driving modes, in particular, the maximum acceleration stops exhibiting optimal results when the train is moving at high speeds.

Newton's Second Law

Conveniently, a rather clear distinction can be made between two approaches to energy-efficient train control. On the one hand, there is research aiming to achieve analytical optimum via use of Pontryagin's Maximum Principle[2] — this route has been discussed in our literature review so far. They then tend to formulate train physics as a case of Newton's Second Law in Equation (1).

$$m\frac{\mathrm{d}v}{\mathrm{d}t} = F - R_b(v) - R_l(x) \tag{1}$$

here, on the left hand side of the equation, m corresponds to the train mass and $\mathrm{d}v/\mathrm{d}t$ is the acceleration of the train. On the right hand side, F corresponds to the force of traction or braking, while $R_b(v)$ and $R_l(x)$ correspond to basic resistance encountered when running at speed v and line resistance at position x, respectively. Basic resistance is usually approximated using the Davis formula[8] in Equation (2).

$$R_b(v) = c_1 + c_2 v + c_3 v^2 \tag{2}$$

here, v is the current speed of the train and c_1, c_2 and c_3 are constants obtained experimentally throughout the simulation. Meanwhile, the line resistance is expressed as in Equation (3).

$$R_l(x) = mg\sin(\theta(x)) \tag{3}$$

here, m is the train mass, g is the gravitational acceleration, and $\theta(x)$ is the slope angle at position x.

This formulation works well with Pontryagin's Maximum Principle due to control actions directly affecting the acceleration of the train, which makes its use convenient for defining optimisation equations and identifying when the train should change between different driving modes.

Law of Conservation of Energy

On the other hand, some studies formulate the motion of the train using the law of conservation of energy in Equation (4).

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(E_p + E_k\right) = W_t - W_b - W_r - W_l \tag{4}$$

here, on the left hand side of the equation $\mathrm{d}(E_p + E_k)\,/\mathrm{d}t$ corresponds to the change in total energy, expressed as the sum of potential energy E_p and kinetic energy E_k. On the right hand side, the components W_t, W_b, W_r, W_l represent the work done by traction and the opposing work of mechanical braking, resistance, and loss, respectively. This approach is often expressed with position as the independent variable, which makes it well suited for modelling routes with varying gradients and elevation profiles. However, in contrast to force-based dynamics, this formulation is less compatible with Pontryagin's Maximum Principle, since control actions affect the system indirectly through speed rather than acceleration. Consequently, energy-based models are better suited for simulations and data-driven methods, where analytical switching conditions are not required.

1.3 Control Optimisation Algorithms

As discussed in earlier sections, Pontryagin's Maximum Principe[2] is used by many researchers as a tool for defining optimal driving modes. However, as Scheepmaker et al.[1] point out, arriving at an optimal sequence of those driving modes is non-trivial. The methods described in different papers vary in how strictly they follow analytical solutions and in how much freedom they allow when modelling train dynamics, operational constraints, and infrastructure characteristics. In their work where Cunillera et al.[9] evaluate the benefits of eco-driving, they also summarise the algorithms used in other articles for achieving energy-efficient train control, drawing distinction between analytical and numerical algorithms.

Analytical Algorithms

One group of such algorithms can be summarised as analytical algorithms, which in turn often rely on Pontryagin's Maximum Principle to derive the necessary conditions for an optimal speed profile. This group mainly consists of constructive and dynamic programming approaches while also including some of more specific approaches such as sequential quadratic programming or mixed-integer linear programming.

Constructive algorithms generate speed trajectories by systematically applying the switching conditions defined by the maximum principle. Although early works such as those by Ichikawa[3] and Howlett et al.[10] model the train with simplified dynamics and piecewise constant track gradients, later research such as that by Khmelnitsky[6], Albrecht et al.[11] and Yang et al.[12] expand the problem to more complex gradient profiles.

Dynamic programming approaches, on the other hand, discretise the state and control space and explore all feasible trajectories to find the optimal solution. The works by Miyatake and Ko[13] and Lu et al.[14], among a selection of other methods, also illustrate how application of dynamic programming along with optimal control theory can aid in deriving an optimal speed profile. This method is particularly useful when constraints, such as speed limits or coasting zones, make purely analytical expressions especially complex.

The sequential quadratic programming method by Miyatake and Ko[13] and the Lagrange multiplier method by Rodrigo et al.[15] have also been applied to discretised versions of the optimal control problem, allowing researchers to optimise speed trajectories while enforcing other complex constraints, such as minimum travel time or passenger comfort.

Another alternative described by Wang et al.[16, 17] is to approximate the non-linear problem as a mixed-integer linear programming problem and then arrive at an optimal solution using existing solvers.

Numerical Algorithms

Although analytical methods are computationally efficient, they require substantial simplifications in train and line models to be applicable. In contrast, numerical methods can include more detailed models without sacrificing accuracy. This group consists mainly of direct search, brute-force, and Monte Carlo simulation methods, although it is constantly expanding with developments in nature-inspired computational intelligence algorithms.

A series of classical techniques such as direct search by Wong and Ho[18], brute-force by Zhao et al.[19], and Monte Carlo simulation by Tian et al.[20] have been examined to explore the solution space when analytical solutions are impractical. Although these methods can handle more complex physics and operational constraints, they tend to either be computationally expensive or fall into local optima without finding the global best solution, particularly when the route is long or the discretisation is fine.

Nature-inspired algorithms have also been extensively considered for eco-driving optimisation. More specifically, Sicre et al.[21], Lu et al.[14] and Li et al.[22] approached the problem through the use of the general formulation of genetic algorithm while extended formulations were also used by some researchers, such as genetic algorithm with fuzzy parameters by Cucala, Sicre et al.[23, 24] and multi-population genetic algorithm by Huang et al.[25] Alternative solutions to energy-efficient train control include simulated annealing as implemented by Keskin and Karamancioglu[26], ant colony optimisation by Lu et al.[14], differential evolution by Kim et al.[27], and similar. As the field of meta-heuristic combinatorial optimisation evolves, even more algorithms suitable for energy-efficient train control are expected to emerge.

Since all of the algorithms mentioned have their own individual advantages and can perform at different levels of quality depending on the formulation of a specific problem, there is no definitive agreement on which is best suited to solve the problem of energy-efficient train control. The choice of method depends on the desired trade-off between model accuracy, computational cost, and adaptability to different routes and operational conditions.

1.4 Reinforcement Learning

The foundations of reinforcement learning can be traced back to the work by Bellman on dynamic programming[28, 29]. Although described as a purely mathematical approach for the evaluation of strategies that act in stochastic environments, these ideas were refined by other researchers into what has now become a separate field under machine learning classification.

As an extension to Bellman's work, Howard[30] introduced Markov decision processes, which provided a structure for sequential decision-making problems under uncertainty. In the context of reinforcement learning, Markov decision processes evolved into a formulation of how the agent interacts with an environment by taking actions and receiving observations and rewards.

A later significant contribution was presented by Barto et al.[31] where they designed a neural-network-based reinforcement learning system that was capable of interpreting rewards from certain actions using actor-critic architecture. Their work showed how action value estimation and policy improvement could be combined to achieve learning behaviour.

The work by Sutton[32] further advanced the field with the introduction of temporal-difference learning, which blended ideas from Monte Carlo evaluation and dynamic programming. Temporal-difference learning showed that agents can incrementally update value functions from successive predictions without waiting for completion of entire trajectory, which allows to speed up learning and stabilise the training process.

Watkins[33] proposed the Q-learning algorithm, which allowed agents to learn optimal action-value functions directly from interactions with the environment. This approach gave more feasibility to exploration of potential routes due to Q-function being updated not for exact trajectory that is being executed, but rather on the optimal policy that is encoded in Q-function up to that point in training process.

In contrast to methods that aim to maximise Q-values and then form the policy that follows the highest value, Williams[34] developed a policy-gradient method, which allowed agents to directly optimise their policies from expected action rewards. This approach allowed to overcome the issues that value-based methods faced with high-dimensional or continuous action spaces, expanding the feasibility of reinforcement learning to a larger set of problems.

1.5 Reinforcement Learning in Rail Transport

Farazi et al.[35] provide a comprehensive survey of how deep reinforcement learning has been applied in transportation research, highlighting the main problem domains, algorithmic decisions, and emerging methodological gaps. Their review analyses applications ranging from traffic signal control and autonomous driving to railway operations, emphasising how deep reinforcement learning enables adaptive, data-driven decision-making in complex and uncertain transport environments.

According to their review, recent studies on automatic train operation systems using deep reinforcement learning focus on minimising energy consumption while satisfying operational constraints such as delay, punctuality, and passenger comfort. Prior work typically represents the speed and position of the train as the state and models acceleration or deceleration as the action, although, the definition of the reward vary. Some studies prioritise energy efficiency alone, whereas others integrate both energy use and delay penalties. More elaborate designs incorporate deviations from target speed profiles, train-to-train spacing and other problem-specific aspects, such as properties of the rail wireless network. Additional research in shunting yard operations expands the state to include arrival and departure conditions, using deep reinforcement learning to optimise parking and dispatch decisions to reduce operational errors.

1.6 Comparison of Reinforcement Learning Solutions in Rail Transport

For a more detailed comparative analysis, we choose two recent works by Shang et al.[36] and Wang et al.[37] where they present their solutions to energy-efficient train control using reinforcement learning. While applying the same underlying principles of reinforcement learning and aiming to solve the same issue, both works enhance the solution with their novel viewpoints on key aspects of implementation such as operational constraints, neural network models, and reward functions. Nevertheless, some of the aspects are effectively matching between the works, such as the state and action definitions for the agent.

State

When describing the state which the agent is in, both Shang et al. and Wang et al. formulate it as a combination of current position along the route, current speed of the train and the time component. However, a minor difference in notation is present where Shang et al. define the time component as the amount of time that has passed since the beginning of the trip and Wang et al. phrase it as the remaining time, both the definitions are effectively equivalent.

Action

The action space is defined in the works by both groups of authors as an adjustment to control force. While Shang et al. alter the control force gradually in select increments to ensure smoother transitions between subsequent acceleration levels, Wang et al. allow any level of control force to be applied in the range from maximum braking to maximum acceleration at any point in the process. Nevertheless, Wang et al. design countermeasures against unreasonable control force selection sequences such as sharp braking after strong acceleration in their reward function. Both Shang et al. and Wang et al. acknowledge the need for discretised action space due to train control limitations.

Operational Constraints

In both works by Shang et al. and Wang et al., constraints related to speed limits, traction and braking force boundaries, and precise stopping at stations are taken into account. Although energy efficiency and adherence to the set schedule are core requirements in problem formulation, in both articles a close attention is paid to passenger comfort requirements. Despite these similarities, the two works differ in how the constraints are enforced during the training of the models. Shang et al. embed the constraints into the environment dynamics, restricting the agent from selecting actions that would violate operational limits. In contrast, Wang et al. allow the agent more freedom during action selection and mainly address constraint violations through penalty terms in the reward function.

Reward Function

In both articles, the reward function is designed to encourage energy-efficient train operation while ensuring that the train arrives at the station on time. Shang et al. and Wang et al. both include energy consumption as a major component of the reward, penalising unnecessary acceleration and inefficient driving patterns. In addition, both works incorporate time-related objectives to prevent excessive delays or early arrivals, reflecting real-world timetable requirements.

However, the structure and emphasis of the reward functions differ between the two approaches. Shang et al. use a relatively compact reward design that does not expand further than the already mentioned requirements for energy efficiency and punctuality. Wang et al., on the other hand, rely more heavily on reward shaping to guide the learning process. Their reward function includes an additional term that motivates high entropy during training for wider exploration, as well as multiple penalty terms to discourage behaviours such as sharp acceleration and deceleration or excessive speeding. Such reward function design choices are strongly correlated with the underlying reinforcement learning system structures employed by the two groups of authors.

System Structure

The system structures proposed by Shang et al. and Wang et al. reflect their different approaches to enforcing constraints and guiding the learning process. Shang et al. design a reinforcement learning model integrated with a reference system that works alongside the learning agent and limits the actions the agent is allowed to take, ensuring that operational constraints are always satisfied during training.

Based on this structure, Shang et al. propose two reinforcement learning algorithms. The first algorithm builds on the standard deep Q-network implementation where the reference system oversees the outcomes proposed by the deep Q-network and may choose to interfere, forcing a certain action for the agent if it could reduce the risk of unfeasible operation. The second algorithm encompasses the deep deterministic policy gradient approach, where the addition of actor-critic architecture allows generation of actions in a continuous action space, while also guiding the decisions through the reference system.

Wang et al., in contrast, adopt a more flexible reinforcement learning system structure. In their approach, entropy is defined as a measure of randomness in action selection and is included directly in the learning objective. By encouraging higher entropy during training, the agent is motivated to explore a wider range of possible control actions instead of rapidly converging to a single strategy. Moreover, the authors employ a combination of two actor-critic network pairs and base the decision of an agent on the network that generates a worse outcome in order to avoid over-fitting. However, the flexibility that the agent is allowed during the training process is strongly countered by a careful design of the reward function.

Summary

Overall, Shang et al. adopt a more conservative and rule-based approach to handling operational constraints, while Wang et al. rely on reward shaping to gradually teach the agent acceptable behaviour. Both approaches aim to balance energy efficiency with safety and comfort, but achieve that using entirely different levels of enforcement during train control simulation.

Although the addition of the reference system by Shang et al. allows for faster convergence during training, it reduces the extent of action space exploration by the system. In contrast, Wang et al. explicitly encourage the system to explore as large an action space as possible so that the reinforcement learning system can derive its own decisions about what actions are favoured by the agent.

1.7 Route Segmentation

Accurately representing the route as a relatively small number of segments can be a valid simplification when considering the reinforcement learning approach to train control. In comparison to analytical methods that benefit from having exact height information in every position along the route, reinforcement learning-based approaches may experience deterioration of performance when decision making logic is performed too often due to the risks of over-fitting the neural networks.

Early studies on optimal-control methods such as those by Ichikawa[3] and Khmelnitsky[6] often used rough, piecewise-constant gradients to represent the train trajectory. Later research showed that splitting the route into smaller segments allows to compute resistance forces more precisely and improves the identification of optimal switching points between driving modes. Howlett and Pudney[38] emphasise that discretising the track into segments aligned with gradient or curvature changes results in more reliable energy estimates without an excessive computational burden.

As a method for track segmentation, Cheng et al.[39] propose a three-stage subdivision. Initially, the track is segmented along slope changes and speed limit changes. Then the larger segments obtained this way are each further subdivided into smaller segments of equal length, bound by some pre-specified maximum length that cannot be exceeded. The process as such ensures that the route segments obtained are flexible enough to accommodate changes in height profile and operational speed requirements while also being of comparable sizes to each other for better reinforcement learning capabilities.

1.8 Height Profile Lookahead

When considering reinforcement learning solutions for energy-efficient train control, environment information such as slope along the route can be conveniently incorporated into the simulation of the selected action by the agent. However, not much literature seems to analyse the effect of providing information about the future height profile in advance, even though such information could influence the current decision of the agent.

The recent study by Merlis[40] derives a theoretical formulation on incorporating lookahead information to help reinforcement learning agents make better decisions in uncertain environments. They show that when some information about the future is known in advance and is not influenced by the agent's actions, this information can be added to the state without breaking the Markov property of Markov decision processes. Having access to knowledge about the upcoming states can enable the agent to plan more effectively and improve the quality of its decisions.

Using a similar idea, Zhang et al.[41] applied this intuition to a real-world railway control problem. Instead of directly supplying future elevation information, they augment deep reinforcement learning with multi-step look-ahead predictions to help trains anticipate future states. Although their approach does not follow the theoretical formalisation by Merlis, the concept of supplying future information into the reinforcement learning model seems to have recently drawn the attention of the researchers. The direct application of the theoretical framework formulated by Merlis could be an interesting direction for future studies in the field of energy-efficient train control.

2 Methods

As with every research, the choice of methodology has direct implications about what possible application areas the work can be extended to as well as how the obtained results should be interpreted. By balancing modelling and simulation approaches between realism and efficiency, we ensure that the study draws focus on solutions that can be well scaled to practical applications. In addition, these methodological choices have a strong impact on how the reinforcement learning solution manages to adhere to its requirements. The way we model the learning agent, its environment, the actions performed, and the rewards obtained determines how well it can understand and adapt to improve energy-efficient driving behaviour over time.

In this chapter, we describe the data preprocessing steps, the physical modelling assumptions, and the algorithmic procedures used to simulate train motion, building a basis for reinforcement learning experiments. We then describe the details of reinforcement learning implementation used to train the agent towards the goal of energy efficiency. In Section 2.1 we present the procedure for constructing the track height profile, including elevation inference, pair-wise distance calculation, and signal smoothing. Section 2.2 introduces the physical model based on the law of conservation of energy, demonstrates how it can be used for train movement simulation and describes the process for the estimation of resistance parameters. Section 2.3 describes the route segmentation procedure, which produces track segments with consistent driving mode, enabling a more stable learning process. The deep reinforcement learning framework is extensively described in Section 2.4, where a strong emphasis is placed on the definitions of the agent, the environment, the state space, the action space, and the reward function. Additionally, a series of design choices are discussed, namely, how operational constraints are imposed on the train simulation, how the neural networks are structured, and the method for height profile lookahead information encoding into the state of the agent. Furthermore, the training procedure is covered with explanations about the interactions between the training loop, the action selection procedure using the epsilon-greedy algorithm, the sampling from experience replay buffer, the policy network parameter updates and the learning rate decay. Finally, in Section 2.5 we discuss how the optimal model is selected after the training loop is performed and in Section 2.6 two methods for hyper-parameter tuning are explained along with a discussion about the use of controlled randomness.

2.1 Track Profile Estimation

Due to the nature of the data in possession, a series of preprocessing steps have to be performed before it can be used in practical computations. The data set consists of a list of 434 geographic coordinates between Kybartai and Paneriai stations. To construct a height profile, along which the train movement simulation is performed, a three-stage process is presented: elevation inference at the coordinate points, distance calculation between these points, and smoothing of the height profile.

Elevation Inference

The height of the track at all locations along the route is resolved using the Google Elevation API[42]. It takes the latitude and longitude values of a selected location and returns the value in metres representing the elevation above the mean sea level for that point. Elevation values are derived from digital elevation models, although their accuracy may vary depending on location and terrain complexity at the particular location. As a result, small imprecisions in the input data can lead to relatively high fluctuations of the height values that are inferred.

Distance Calculation

Given a pair of coordinates with latitude and longitude values, the distance between these points on the surface of a sphere can be derived from the haversine formula. Its practical application for calculating distances between locations was extensively discussed by Sinnott[43], although the formula itself is directly derived from spherical trigonometry using a series of calculations in Equations (5) to (7).

$$d\left(p_1, p_2\right) = R \cdot c\left(p_1, p_2\right) \tag{5}$$

here, $d\left(p_1, p_2\right)$ is the distance between points p_1 and p_2 on the surface of the sphere, R corresponds to the radius of the sphere, which in this case is the radius of the Earth $R \approx 6,371\ \mathrm{km}$ and $c\left(p_1, p_2\right)$ is the central angle between these two points. In turn, central angle between two points is defined in Equation (6).

$$c\left(p_1, p_2\right) = 2 \cdot \operatorname{atan2}\left(\sqrt{\operatorname{hav} \theta(p_1, p_2)}, \sqrt{1 - \operatorname{hav} \theta(p_1, p_2)}\right) \tag{6}$$

here, $\operatorname{atan2}$ is the two-argument arctangent function and $\operatorname{hav} \theta(p_1, p_2)$ is the haversine of an angle $\theta(p_1, p_2)$ between two points p_1 and p_2 on the surface of the sphere, as expressed in Equation (7).

$$\operatorname{hav} \theta(p_1, p_2) = \sin^2\left(\frac{\phi(p_2) - \phi(p_1)}{2}\right) + \cos\phi(p_1)\cos\phi(p_2)\sin^2\left(\frac{\lambda(p_2) - \lambda(p_1)}{2}\right) \tag{7}$$

here, $\phi(p)$ and $\lambda(p)$ correspond to latitude and longitude of a point p, respectively, of two points p_1 and p_2 on the surface of the sphere.

Applying the combination of these three formulas allows us to calculate the distance between two subsequent points along the route. Adding together the pair-wise distances between each two subsequent locations results in a complete route length that the train had to travel from the starting station to the destination.

Elevation Smoothing

Since the elevation values acquired from the Google API tend to contain significant fluctuations, we select the Nadaraya-Watson[44, 45] kernel regression method for noisy signal smoothing using the Gaussian kernel. The illustration of the method is displayed in Figure 2 from the interactive book by García-Portugués[46], where the Gaussian distribution depicted in grey along the x-axis represents the relative weight that the data points at those coordinate x values have when calculating the estimate for the target point in blue. Not only does this method aid in reducing the effect of outliers caused by imprecise location measurements, but it also provides a method for interpolating values for unobserved locations.

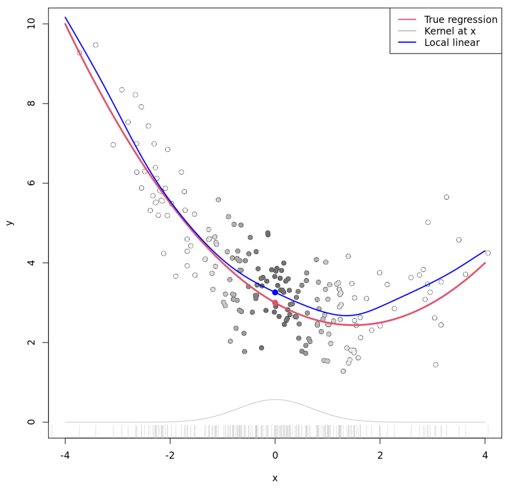


Fig. 2. Nadaraya-Watson[44, 45] kernel regression using Gaussian kernel

The Gaussian kernel weights for the observed data points are derived from the distance between a selected location and those respective points using Equation (8).

$$w_i = \exp\left(-\frac{\text{dist}(x, x_i)^2}{2\sigma^2}\right) \tag{8}$$

here, w_i is the weight corresponding to an observed data point x_i when calculating the smoothed value for point x. The distance measure $\text{dist}(x, x_i)$ between the points x and x_i can be domain specific, although in our calculations we use the haversine distance described previously. The term σ corresponds to the standard deviation of the Gaussian distribution with lower values resulting in strong emphasis toward closer data points, while higher values produce a wider smoothing effect. The smoothed elevation value for the point x along the route can then be approximated as a weighted average of the elevation values of the observed data points using Equation (9).

$$h_{\text{smooth}} = \frac{\sum_{i \in I} w_i h_i}{\sum_{i \in I} w_i} \tag{9}$$

here, h_{smooth} is the smoothed elevation value of a point x, with weights w_i calculated in Equation (8) and h_i corresponding to the height values of each observed data point x_i from a complete set of points I along the route.

2.2 Train Movement Simulation

By applying a theoretically sound simulation of the train movement, we acquire the means to predict how particular actions during train operation affect its dynamics. In turn, formulating the law of conservation of energy enables modelling the train motion along the track with uneven elevation. When all work and energy contributions acting on the train over a track segment are formulated into a single energy balance equation that sums to zero, this balance can be approximated numerically, allowing to estimate the speed at which the train exits the route segment, given all the known and already estimated parameters and operational conditions.

Law of Conservation of Energy

Assuming that the train is a point mass, moving along the segment of the route, we express the balance of work done by different elements of the system in Equation (10).

$$W_{\text{eng}} = \Delta E_{\text{pot}} + \Delta E_{\text{kin}} + W_{\text{fric}} + W_{\text{air}} \tag{10}$$

here, W_{eng} is the work done by the train engine, while ΔE_{pot} and ΔE_{kin} are changes in the potential and kinetic energy of the train, respectively, and W_{fric} and W_{air} correspond to the work done by friction and air resistance, respectively. Individually, each component is expressed in Equations (11) to (15).

$$W_{\text{eng}} = P\frac{\Delta S}{\bar{v}} \tag{11}$$

here, P is the power at which the engine is running and $\Delta S/\bar{v}$ is the running time, for convenience expressed as the ratio between the distance travelled ΔS and the average speed $\bar{v}$ throughout the segment.

$$\Delta E_{\text{pot}} = mg\Delta h \tag{12}$$

here, m is the mass of the train, $g = 9.81\,\mathrm{m/s}^2$ is the gravitational acceleration, and Δh is the change in elevation throughout the segment.

$$\Delta E_{\text{kin}} = \frac{1}{2}m\left(v_{\text{end}}^2 - v_{\text{start}}^2\right) \tag{13}$$

here, m is the mass of the train and v_{start} and v_{end} correspond to the speed of the train at the start and at the end of the segment, respectively.

$$W_{\text{fric}} = \mu mg\Delta S \tag{14}$$

here, μ is the friction coefficient, m is the mass of the train, $g = 9.81\,\mathrm{m/s}^2$ is the gravitational acceleration, and ΔS is the distance travelled throughout the segment.

$$W_{\text{air}} = \frac{1}{2}\rho C_d A\bar{v}^2\Delta S \tag{15}$$

here, $\rho = 1.225\text{kg/m}^3$ corresponds to the air density at standard atmospheric conditions, C_d is the drag coefficient, A is the cross-sectional area of the train, $\bar{v}$ is the average train speed, and ΔS is the distance travelled throughout the segment.

When combined together, these equations fully describe the energy state of the train, enabling the simulation of train movement over the known landscape with varying power levels.

Stepwise Simulation

Since the tracks are laid on an uneven landscape, which we estimate using a smoothed height profile h_{smooth}, the train movement simulation cannot be performed in a single calculation step. In order to follow the dynamics of the train more accurately, whenever running the movement simulation for a route segment, we further subdivide it into small steps dS within which we assume linear conditions, namely, that the track is a straight line, as illustrated in Figure 3, and that the speed changes evenly from v_{start} to v_{end}, resulting in the average speed approximation of $\bar{v} = (v_{\text{start}} + v_{\text{end}})/2$. With such conditions, the actual distance travelled by the train can be expressed using the Pythagoras theorem between the horizontal distance covered and the change in elevation, as expressed in Equation (16).

$$dS_{\text{diagonal}} = \sqrt{dS^2 + dh^2} \tag{16}$$

here, dS_{diagonal} corresponds to the distance travelled by the train along the uneven landscape, taking into account the horizontal distance dS and the change in elevation dh.

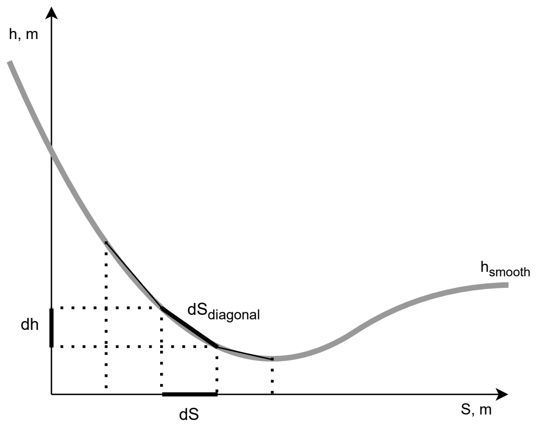


Fig. 3. Track profile discretisation to steps dS

As we initialise the simulation with known information, including initial train speed, train mass, engine power level, route step length, its change in elevation, and other resistance parameters, the only variable missing from the energy balance equation is the terminal speed at the end of the route step. Therefore, by using numerical approximation methods, we can find the terminal speed, fully defining the speed profile at which the train is moving at every step.

Additionally, we can estimate the energy consumption incurred by the train during each such step by substituting the approximated average speed $\bar{v}$ into the physical expression in Equation (11) to derive a practical approximation in Equation (17).

$$W_{\text{eng}} = P \frac{2 \cdot dS_{\text{diagonal}}}{v_{\text{start}} + v_{\text{end}}} \tag{17}$$

here, W_{eng} corresponds to the work performed by the train engine, P represents the power level at which the engine is operating, dS_{diagonal} is the distance travelled by the train along uneven terrain, and v_{start} and v_{end} correspond to the speed at which the train enters and exits the segment, respectively. As a result, by discretising the route into small steps within which linearity assumption is feasible, one can estimate the energy consumption of the train engine throughout each such step, and thus throughout the entire trip.

Resistance Parameter Estimation

The resistance parameter values are estimated by performing a grid search, as described in Section 2.6, while keeping parameters A and m fixed and parameters μ and C_d varying in a pre-specified range. For each combination of parameters, a train movement simulation is performed in segments corresponding to the distance between the locations provided in the data set. At the start of each such segment, the movement simulation is initialised with speed and power values from the data. The estimated train speed at the end of the segment is compared with the actual speed taken from the data, as illustrated in Figure 4, and the mean squared error is calculated using Equation (18).

$$\epsilon_{\text{MSE}} = \sqrt{\frac{1}{|I|} \sum_{i \in I} \left(v_{\text{true},i} - v_{\text{est},i}\right)^2} \tag{18}$$

here, ϵ_{MSE} is the calculated error metric for a particular combination of resistance parameters, I is the set of all points in the data set, $v_{\text{true},i}$ corresponds to the actual train speed at the end of the segment from i to $i+1$, as recorded in the data, and $v_{\text{est},i}$ is the estimated speed at the end of the segment. The set of parameter values μ and C_d that reaches the lowest value of the error metric is considered to be optimal.

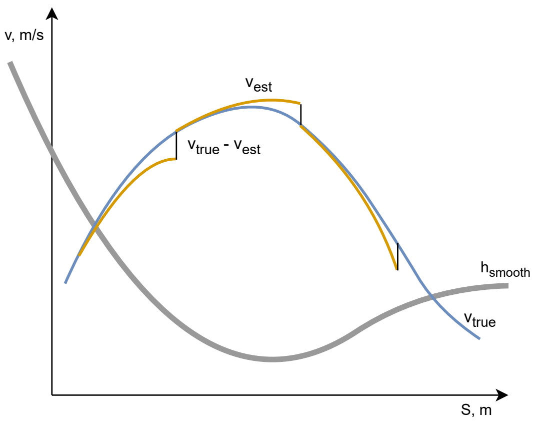


Fig. 4. Movement simulation parameter estimation from segment terminal velocity

2.3 Route Segmentation

Subdividing the route into segments of comparable lengths can improve the performance of the reinforcement learning application by allowing for a more consistent simulation of environment actions. A simple approach of dividing the route into segments of fixed length is considered as one of the approaches to route segmentation. However, it tends to result in a loss of important track properties, such as slope and speed limit requirements, which may change multiple times throughout a single longer segment.

A concise set of engine power switching points, as proposed by Cheng et al.[39], can be derived from the observed changes in route elevation and the limited speed regions along the track. However, deciding the segments of constant slope is non-trivial. For this task, we propose a heuristic algorithm that constructs the boundary points at which the slope changes occur. These segments are then combined with the requirements imposed by the speed profile, and all the combined segments are discretised with a maximum segment length requirement to produce a list of segments whose lengths are less varying in size between one another. Finally, we optionally add a small amount of Gaussian noise to the obtained route segment boundaries to better accommodate the imprecisions with which the train operator performs actions.

Slope Change Detection

Having calculated a smoothed height profile of the route $h_{\text{smooth}}(x)$, we define an approximate slope h' at each step multiple of ΔS along the route in Equation (19).

$$h'(k\Delta S) = \frac{h_{\text{smooth}}(k\Delta S) - h_{\text{smooth}}((k-1)\Delta S)}{\Delta S} \tag{19}$$

here, $h'(k\Delta S)$ is the numerically estimated slope of the track at position $k\Delta S$, such that k acquires integer values from 1 to $\lfloor S_{\text{total}}/\Delta S \rfloor$ inclusive, and ΔS is the selected step precision for calculations.

Throughout the upcoming explanation, please refer to Figure 18 in Section 3.4 for an illustrative example. We define the slope toleration threshold h'_{thresh} and iterate over every calculated value $h'(k\Delta S)$ to find locations along the route where the slope values transition over the threshold h'_{thresh}, considering both positive slope values with positive threshold h'_{thresh} and negative slope values with negative threshold $-h'_{\text{thresh}}$. We save these locations with labels $\text{dir}(k) = +1$ or $\text{dir}(k) = -1$, indicating in which direction the slope exceeded the threshold. We then iterate over the list of indicators and group all crossings that are subsequent and signify the same direction. Whenever a position is encountered where the direction of the slope changes, this position marks the end of the previous segment and the start of the new one.

Limited Speed Segments

To ensure safe operation, the speed of the train is regulated throughout the entire route with what is defined as the speed profile. Although there is a default maximum speed limit that must never be exceeded throughout the track, an additional set of requirements enforce even lower maximum speed limits for the track segments that are usually positioned around the train stations. Due to the fact that different segments have different maximum speed requirements, these requirements are expected to shape the train operation strategy. As a result, each entry and exit of the limited speed zone should be included as the defining boundary for the route segmentation task.

Combined Segment Discretisation

With individual boundaries defined through slope changes and maximum speed limit changes, we combine the two sets of requirements and build a single schema for when the train operation strategy can be adjusted. We also further subdivide segments that are longer than a pre-specified distance value ΔS_{max} into a set of smaller segments of equal length that no longer exceed ΔS_{max}. Consequently, the segments received from the procedure tend to be of more similar sizes to each other, suggesting a more stable reinforcement learning process.

Imitating Action Imprecisions

Once the route segmentation is constructed in terms of segment switching boundaries, we add a small amount of Gaussian noise to each of them, as expressed in Equation (20).

$$x_{i,\text{noisy}} \sim x_i + \mathcal{N}\left(\mu, \sigma_{\text{seg}}\right) \tag{20}$$

here, $x_{i,\text{noisy}}$ correspond to the route boundary points acquired from the original boundary point x_i by adding a Gaussian noise $\mathcal{N}$ centred around $\mu = 0.0$ with the standard deviation σ_{seg} expressed in metres. The definition of imprecise segment switching points serves two main purposes. From a practical standpoint, such adjustment imitates the imprecise actions performed by the train operator, where train power switching could be done either in advance or with a delay. In terms of implementation specifics, the added noise increases the amount of possible state representations for the neural network training within the reinforcement learning implementation, resulting in a better generalisation for different possible environment states.

2.4 Deep Reinforcement Learning

The deep reinforcement learning application in our work relies on neural networks to represent the policy according to which the agent chooses its actions at different states. The learning agent repeatedly interacts with a simulated environment, observes the current state, selects an action, and receives a reward that reflects the quality of that action, as illustrated in Figure 5 taken from the book by Sutton and Barto[47]. Model training is performed over many episodes during which the network parameters are updated using gradient-based optimisation. To improve training stability and efficiency, we employ common techniques, such as experience replay and a separate target network. The components of network architecture, state representations, and reward design are described in this section in general terms, although the process of arriving at a suitable set of parameter values is described in more detail in Chapter 3.

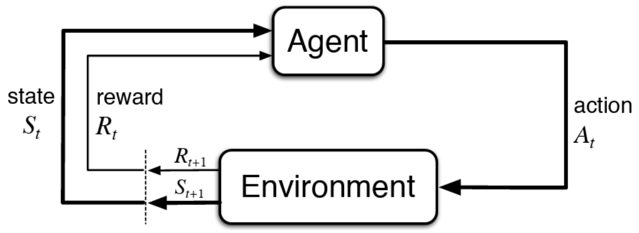


Fig. 5. Reinforcement learning scheme by Sutton and Barto[47]

Environment

In reinforcement learning applications, the environment encompasses everything the learning agent interacts with. It includes the laws of physics that govern the train movement simulation and describes how the viewpoint of the learning agent changes throughout the simulation.

The train movement simulation procedure, as described in Section 2.2, acts as the core part of the interaction with the environment in the context of reinforcement learning applications. It converts the action requests made by the agent into information about the next state the agent encounters. In addition, the environment is responsible for providing feedback to the agent after the interaction is performed. This feedback is problem specific and is shaped through the reward function, as discussed in Section 2.4. Notably, the reward is expected to be formulated in terms of the properties and encounters that were present during the interaction of the agent with the environment. Otherwise, the agent might fail to discern the relationship between the action chosen and the reward received, resulting in an unstable learning process.

State Space

The state of the agent is a set of dimensions that can be interpreted as a snapshot of the environment at a specific moment. In the context of deep reinforcement learning, the state acts as an input to the neural network which is tasked with navigating the agent to its next state. The neural network can be implemented to either approximate the Q-values of the next state or directly propose a new action via the policy gradient approach. In this work, only the implementation of the policy gradient is considered.

The minimum necessary state of the train consists only of physical values that are needed for movement simulation, namely, the current location along the track S, the current speed v, and the time spent until the current moment t. However, since these values are supplied directly into the neural network, we tend to modify them with the goal of improving the performance of the network. The exact modifications will be further discussed in Chapter 3, although two main methods can be discerned, which are not mutually exclusive.

- Regularisation of state dimension values is arguably the most important modification to the state representation. Since the state is often expressed as a set of dimensions with vastly different units of measure and orders of magnitude, the imbalance as such usually prevents the network parameters from discovering meaningful patterns. Although careful selection of learning hyperparameters and network structure could alleviate the issue, the usual approach in the industry is to avoid this complication altogether. Thus, before supplying the values to the network, it is a common practice to rescale the interval of possible values to either a fixed range or to apply standardisation, ensuring comparable magnitudes across state dimensions and improving numerical stability during training.
- Augmentation with auxiliary values is a problem-specific approach that could enhance the performance of the model. Apart from the strictly necessary dimensions needed for the physical simulation, one may choose to include a selection of derived properties that are expected to provide additional information for the model. The reinforcement learning applications often exhibit the behaviour of over-fitting, which may or may not be desirable, depending on the specifics of the problem. The state augmentation can act in two ways — either facilitate the learning process of the model in the currently existing problem scope or it can expand the scope for the application of the model without degrading its performance.

Action Space

We define the action space as a selection of power level for train engine operation, which is assumed to be a discretised value due to practical limitations of train control. In the problem formulation, we define power values from $P_{\min} = 0\,\mathrm{kW}$ to $P_{\max} = 1600\,\mathrm{kW}$ divided into $P_{\mathrm{count}} = 9$ equally spaced power levels in increments of $\Delta P = (P_{\max} - P_{\min}) \,/\, (P_{\mathrm{count}} - 1) = 200\,\mathrm{kW}$. Moreover, once the agent selects the power level, it remains constant throughout the entire duration of the route segment. Notably, we choose not to consider braking as a possible action due to the goal of energy efficiency and a complicated interaction between braking and engine power setting. For freight trains without installed regenerative braking systems, every instance of unnecessary braking directly translates into wasted energy consumption as a consequence of poor planning.

Operational Constraints

As discussed so far, the learning agent is capable of interacting with the environment by selecting particular actions and transitioning between states. However, if they are not restrained, they could violate some of the restrictions imposed on train operation. For instance, maximum speed limitations, as previously described in Section 2.3, must also be ensured in any valid solution to energy-efficient train control. The practical methods for following the requirements vary in their severity levels and are tightly coupled with the design choices for the reward functions. We make a distinction between three possible strategies for balancing operational constraint enforcement with reward shaping.

- Early termination is one of the potential approaches to handling violations of formal requirements. In the scenario where the agent enters an invalid state during the training process, the training becomes immediately interrupted, and an optional penalty term is assigned to ensure that the cumulative reward for the episode is sufficiently punishing for the agent to learn to avoid such outcome.
- Non-terminating penalty is a less strict approach for discouraging the agent from unwanted behaviour. Since the training episode does not end after the violation is encountered, this method requires carefully balanced rewards to make sure that the agent is not learning how to abuse the violations.
- Enforcement of the desired behaviour is an alternative method aimed at preventing violations. Instead of shaping the behaviour of the agent via feedback from the environment, the choice is made to always avoid unwanted behaviour from happening in the simulation of the action. However, unless other sources of feedback are involved, the agent may learn to ignore the requirement entirely, resulting in worse scalability to practical applications.

In our work, we are faced with two sets of operational constraints. The train is expected to arrive at the destination no later than a set time target $t_{\max}$ and is not allowed to violate speed restrictions anywhere along the route. Furthermore, we not only aim to conform to the set maximum speed limits, but also self-impose a minimum speed requirement $v_{\min} = 1\ \mathrm{m/s}$ as a safety measure. It acts as a buffer, preventing the train from stopping completely in cases when the mismatch between the simulation and the real-world train movement becomes unexpectedly high.

In our work, we choose to employ following methods to ensure that the operational requirements are satisfied:

- The time target is not enforced explicitly, but encouraged with high value rewards.

- The maximum speed limit $v_{\max}$ is enforced in the train movement simulation. This design choice has two practical consequences:
  - If the agent selects the power level that results in the simulated train speed exceeding a maximum value $v_{\max}$, then the simulated train speed is truncated to the exact value of $v_{\max}$. As a result, excess power becomes effectively wasted and this behaviour becomes discouraged by the reward function that promotes energy efficiency.
  - Whenever the train is about to violate the maximum speed limit $v_{\max}$, the operator person is expected to interfere and manually slow down the train to an approximate speed of $v_{\max}$. This type of intervention acts as both a speed limit enforcement and an adjustment of the actual movement of the train to match the simulation process, which internally also truncates the speed value to $v_{\max}$.
- The minimum speed limit $v_{\min} = 1\ \mathrm{m/s}$ is encouraged in a combination of two approaches:
  - The train movement simulation enforces the speed value to always be at least $v = 0.1\mathrm{m/s}$. By preventing the speed from reaching $0\ \mathrm{m/s}$, we ensure that the denominator in Equation (17) is greater than zero and the expression does not approach infinity.
  - Whenever the speed of the train falls below $v_{\min} = 1\ \mathrm{m/s}$, the agent gets a penalty, thus discouraging them from violating the minimum speed limit.

Reward Function

Throughout the work, we consider two definitions of the reward function that aim to achieve slightly different goals. One reward function focusses solely on reaching the set time target, while the upgraded version of the reward function also incorporates the aspect of energy efficiency. The structure of both functions follows the same pattern, comprising of three components.

- Base reward incentivises the agent to follow the main objective while performing the actions in the preferred way. As the agent selects actions that are closer to the expected behaviour, they are assigned higher reward values, this way receiving feedback regularly.
- Punishments are deducted from the reward points when the agent makes mistakes or acts in an undesired way, thus discouraging poor or risky actions. Depending on the problem formulation, the punishments can either be avoided entirely or are expected to be balanced with other components of the reward function to achieve the maximum cumulative reward over the episode. Since some of the operational constraints are not enforced but rather shaped through the reward function design, balancing the punishment effects is particularly important for ensuring that they are conformed to.
- Reward on success acts as an extra bonus given when the agent completes the task correctly within the desired limits. It motivates the agent to finish the task properly and is supposed to dominate the expression of the discounted reward function at each step.

The reward discounting approach is often applied to balance immediate rewards with future rewards, allowing the agent to better anticipate the long-term effects of the selected action. This is achieved by introducing a discount factor γ, which exponentially reduces the weight of the rewards received in future time steps, as expressed in Equation (21).

$$G_t = R_t + \gamma \cdot G_{t+1} \tag{21}$$

here, G and R correspond to discounted and immediate rewards, respectively, while the subscript t indicates the time step within the training episode. To avoid the recursive definition, upcoming discounted reward can be repeatedly substituted with the same expression to obtain an infinite exponential sum in Equation (22).

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i} \tag{22}$$

Discounting ensures that the agent not only aims for short-term gains but also plans ahead to maximise the cumulative reward throughout the episode.

Deep Neural Network Design

The deep reinforcement learning approach used in this work implements two neural networks, namely, a policy network and a target network. The agent uses the policy network to select actions based on the current state, while the target network is introduced to support stable training. Both networks have the same structure, but their parameters are updated in different ways during training.

The network architecture is a fully connected feed-forward neural network, as illustrated in Figure 6. It consists of an input layer, two hidden layers, and an output layer. The input dimension is determined by the size of the state representation, while the output dimension corresponds to the number of available actions. Each hidden layer contains a fixed number of neurons, referred to as the hidden dimension, and all layers are fully connected. As discussed in the state space description, different state representations were tested in this work, requiring the input dimensions of the neural networks to be adjusted for each different state structure. In contrast, hidden dimensions and output dimensions were kept stable with 128 neurons assigned to each of the hidden layers and 9 action choices corresponding to the output dimension, as indicated by $P_{\text{count}} = 9$ during action space definition.

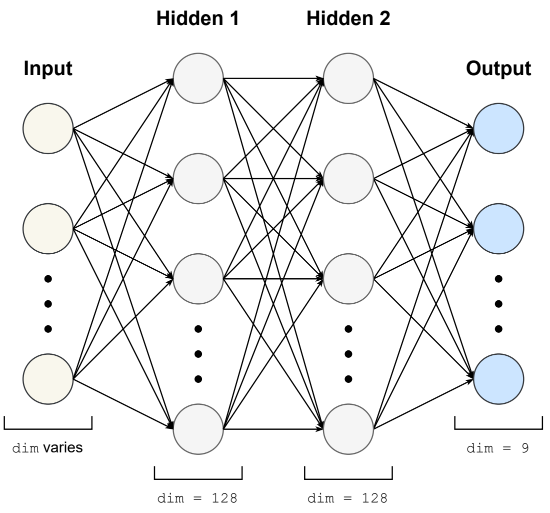


Fig. 6. Neural network scheme

Regularisation techniques were also considered during the network design. In particular, dropout was optionally applied between hidden layers, allowing the network to be trained both with and without dropout regularisation. In addition, two methods were explored for the update process of the target network. The first method involved applying periodic updates to the target network by copying all the parameters from the policy network every 20 episodes. The second method used soft updates, where the target parameters were gradually updated at each episode from the policy network parameters via parameter τ, as expressed in Equation (23).

$$\theta'_i = \tau\theta_i + (1-\tau)\theta'_{i-1} \tag{23}$$

here, θ and θ' correspond to the sets of parameters from the policy network and the target network, respectively, while the subscript i indicates the episode at which the parameter update is performed.

Height Profile Lookahead Encoding

As a direction to improve the decision-making of the learning agent, we propose to include additional information about the upcoming track profile in the state representation. Since train energy consumption is strongly affected by track gradients and speed limits ahead, lookahead encoding allows the agent to anticipate future conditions instead of reacting only to the current state. The three different encoding strategies that were considered in this work are as follows.

- Discrete slope sampling from the smoothed height profile h_{smooth} at M intervals of fixed length l, as displayed in Figure 7, is the most simplistic method proposed to improve the predictive performance of the model. Although this approach gives the agent a coarse view of the upcoming elevation patterns, it lacks the ability to adapt to different lengths of segments for which the action selection procedure is performed.

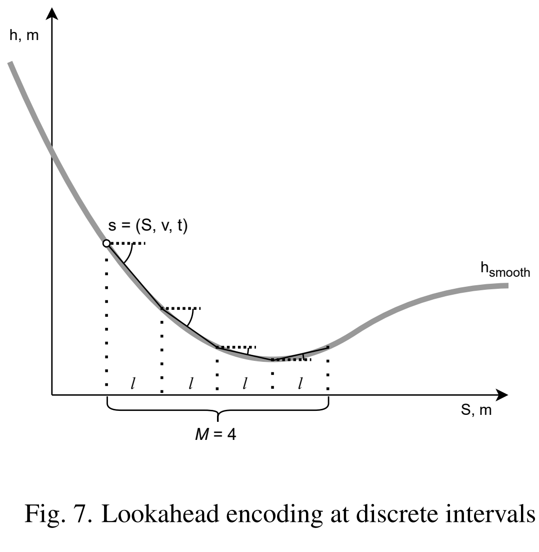


Fig. 7. Lookahead encoding at discrete intervals

- Fractional slope sampling method is formulated as a modification to its discrete sampling counterpart, as illustrated in Figure 8, accommodating to the effect of unevenly sized route segments. This method encodes the elevation differences observed over fractional parts of two segments, one that is upcoming immediately and one afterward. Since the agent makes its decision for the entire duration of the route segment, supplying slope information about individual parts of the segment, such as one third at the beginning, one third at the middle and one third at the end, as well as a lookahead into the first one third of the next segment, can help to meaningfully shape the choice of action.

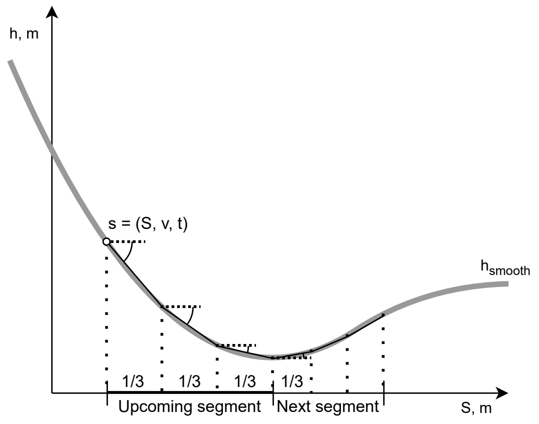


Fig. 8. Lookahead encoding at fractional intervals of segment size

- Segment metadata encoding is built on top of the proposed fractional sampling by including core information about the segments themselves. This improvement is motivated by the fundamental difference between discrete and fractional sampling methods. In particular, since segments can be of different lengths at different locations throughout the route, fractional sampling, in turn, also encodes information about intervals of varying lengths. For the neural network that is tasked with selecting an action, the presence of such uncertainty can result in sub-optimal decisions being made. To counteract this, segment metadata encoding augments the state with additional information about route segment lengths themselves. In addition, to promote better informed decisions, the maximum speed limits for these segments are also included in the state representation.

Although the height profile lookahead encoding can help to convey more information about the upcoming conditions of the track, they also carry the risk of introducing redundant or excessive data. Too many similar inputs can dilute the impact of core features, as training gradients are spread across many dimensions. This may slow down learning, reduce stability, or lead to sub-optimal policies. Therefore, balancing the amount of lookahead information while avoiding unnecessary redundancy is a separate consideration of significant importance.

Training Loop

The training loop is the process during which the agent improves its policy through repeated interactions with the environment. Given a pre-specified number of episodes, the train movement simulation is run during each episode, while allowing the agent to make decisions for what action to take at each segment of the route. The entire training process if clearly summarised in Figure 9 by Kuprikov et al.[48], where purple arrows indicate the loop in which the agent interacts with the environment, green arrows demonstrate how the transitions are formed for future training, and blue arrows correspond to the training process while specifying at what stages are the transition components from the replay memory referenced.

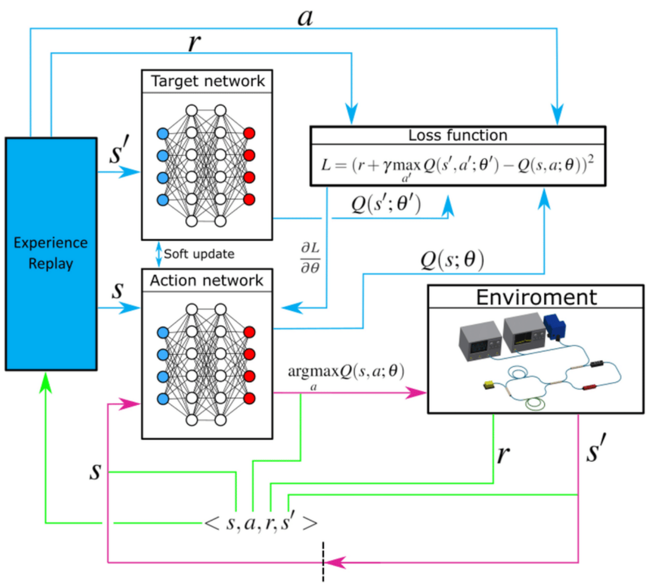


Fig. 9. Training process scheme by Kuprikov et al.[48]

Epsilon-Greedy Algorithm

Initially, the agent makes the action selection randomly by sampling a uniform distribution of all possible actions. However, as training progresses, the agent is more often required to pick the next action that they think is optimal given their current state. This balancing is achieved using the epsilon-greedy algorithm, in which the hyper-parameter ϵ starts at the value of $\epsilon_{\max}$ and decays exponentially toward zero as training progresses. At each step, the agent has the probability ϵ to explore a random action, and the complement probability $1 - \epsilon$ to exploit their already attained knowledge by selecting the action they have learnt so far to be the best. We choose to define the decay rate of ϵ through the parameter $\epsilon_{\text{halflives\_count}}$ that defines how many times the value of ϵ gets halved in size throughout the training loop. Effectively, the value for ϵ at each episode can be expressed as in Equation (24).

$$\epsilon_k = \epsilon_{\max} \cdot 2^{-\frac{k}{t^{\epsilon}_{1/2}}} \tag{24}$$

here, ϵ_k is the rate of random action selection during episode k, $\epsilon_{\max}$ corresponds to the initial value of ϵ and $t^{\epsilon}_{1/2}$ is the half-life at which ϵ decays, as expressed in Equation (25).

$$t^{\epsilon}_{1/2} = \frac{N}{\epsilon_{\text{halflives\_count}}} \tag{25}$$

here, N is the number of episodes that the training loop contains and $\epsilon_{\text{halflives\_count}}$ is the parameter, indicating the number of half-lives that happen throughout the training loop.

Replay Buffer

As the training loop progresses, the replay buffer gets updated with the latest transition records (s, a, s', r) containing the current state s, the selected action a, the next state s' and the reward r obtained at each step of the environment. Initially, while the buffer is yet to be filled with enough entries to form a batch, it only collects information about new transitions and is not used for reading from. Once the warm-up is complete, read requests can be made to the replay buffer, where a number of random transitions are selected to form a batch for policy network training. Once the buffer is filled to its maximum capacity, it starts to drop the oldest entries, effectively acting as a ring buffer that keeps only the most recent transitions, ensuring relevance of the data used in the training process of the policy network.

Network Training

After every environment step, a batch request is made to the experience replay buffer to sample a number of transition records for the policy network training. Similarly to the definition of discounted rewards in Equation (21), the policy network is expected to approximate a Q-function that involves the immediate reward from the action added with the future expected Q-values from a learned policy, as formulated by Bellman[29] in Equation (26).

$$Q^{\pi}(s,a) = r + \gamma Q^{\pi}(s', \pi(s')) \tag{26}$$

here, $Q^{\pi}(s,a)$ is the Q-function corresponding to policy π that approximates the Q-values when action a from state s is taken, r stands for the immediate reward from the interaction, γ is the discount factor and $Q^{\pi}(s', \pi(s'))$ is the approximated reward from the Q-function when following the policy π from the next state s'. Since both sides of the equation contain the same function Q^{π} that is getting approximated at different states s and s', the learning may become unstable. This is alleviated by using the target network with periodic or soft updates from the policy network, as expressed in Equation (23). The error corresponding to the Bellman Equation (26) is defined as the difference between the two sides of the equality and is called temporal difference δ as defined in Equation (27).

$$\delta = Q^{\pi}(s,a) - \left(r + \gamma \max_{a'} Q^{\pi}(s', a')\right) \tag{27}$$

here, the same notation applies as in Equation (26) with an addition of a' being the action taken at the upcoming state s', which is chosen to maximise the Q-function value $Q^{\pi}(s', a')$.

We use the Huber loss function, which acts as a mean squared error for values smaller in magnitude than 1 and as a mean absolute error for larger values, as defined in Equation (28).

$$\mathcal{L}(\delta) = \begin{cases} \frac{1}{2}\delta^2, & \text{for } |\delta| \le 1, \\ |\delta| - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{28}$$

When training the policy network, we supply a batch of transition samples from the experience replay buffer. Therefore, the loss at each environment step is expressed as the average of the Huber loss throughout all the samples in a batch, as defined in Equation (29).

$$\mathcal{L} = \frac{1}{|B|} \sum_{(s,a,s',r) \in B} \mathcal{L}(\delta) \tag{29}$$

here, B corresponds to a batch of replay memory samples (s, a, s', r), randomly selected from the replay memory. With loss values calculated, the parameters of the policy network are updated via gradient descent by moving in the direction of the negative gradient of the loss, with the learning rate determining the step size of the update.

Learning Rate

To ensure stable convergence, the learning rate is not kept constant throughout the training but instead decays over time. At early stages of training, a larger learning rate allows the policy network to adapt quickly, while at later stages a smaller learning rate enables finer adjustments around a near-optimal solution. Similarly to the decay of the exploration parameter ϵ, the decay of the learning rate is defined using the initial learning rate $\alpha_{\max}$ and the parameter $\alpha_{\text{halflives\_count}}$, representing the number of times the learning rate gets halved in size throughout the training loop. Thus, in each training episode the learning rate α can be expressed as in Equation (30).

$$\alpha_k = \alpha_{\max} \cdot 2^{-\frac{k}{t^{\alpha}_{1/2}}} \tag{30}$$

here, k is the number of the episode and $t^{\alpha}_{1/2}$ represents the half-life of the learning rate, defined in Equation (31).

$$t^{\alpha}_{1/2} = \frac{N}{\alpha_{\text{halflives\_count}}} \tag{31}$$

here, N is the number of episodes that the training loop contains and $\alpha_{\text{halflives\_count}}$ is the parameter, indicating the number of half-lives that elapse throughout the training loop.

2.5 Optimal Model Selection

Instead of taking the last model at the end of the training loop to define the policy, we build a metric R^* that balances the result quality and stability of the intermediate results during the training loop, as defined in Equation (32).

$$R^*_i = R_i + \bar{R}^{(w)}_i \tag{32}$$

here, R^*_i corresponds to the derived metric at episode i, R_i is the actual total reward value obtained during episode i and $\bar{R}^{(w)}_i$ is the moving average of the reward values that uses window length w and is centred around episode i. While selecting a model that achieves the overall best reward value could be a reasonable choice, there is a risk that such model might act unstably when encountering fluctuations of the random components throughout the simulation.

Whenever the agent acts during the training loop, it uses the randomness encoded in epsilon-greedy algorithm, which may lead to bad quality decisions being generated, thus producing worse reward values than what is already learned by the policy network. For accurately evaluating the policy that is learned throughout the training loop, a regular evaluation run is included every 10 episodes of training, during which the agent follows only the policy learned, without making any exploratory choices along the way. At the end of the training loop, an iteration over all the rewards obtained in evaluation runs is performed and the model is selected such that receives the highest value of the metric R^*_i. With the moving average window of length $w = 10$, we effectively average out the evaluation run results from 100 episodes of training. Notably, since the reward from the evaluation run i is the component R_i itself and is also included in the moving average $\bar{R}^{(w)}_i$, the reward of this episode has the weight of 0.55 in the final metric R^*_i, while other 9 neighbouring evaluation runs receive the weight of 0.05.

2.6 Hyper-Parameter Tuning

Throughout the work, a large number of parameters may influence how the calculations are being performed and what results are achieved in relation to that. In order to optimise the behaviour of such calculations, we employ two types of hyper-parameter tuning techniques, which allow us to arrive at a set of values that produce optimal results.

Grid Search

When a set of parameters is concise and clearly defined in a range of possible values, we employ the grid search technique. It exhaustively iterates over all possible sets of parameter values from a pre-defined range and evaluates the outcomes achieved by using each set. The set with the best performance is selected as the optimal set for further calculations. Since this method performs computations exhaustively, it suffers from the curse of dimensionality, where the possible number of permutations of parameters increases exponentially with every new parameter added. As a result, the grid search is feasible only for minor sub-tasks with a small set of parameters from a clearly defined range of possible values.

Iterative Feedback Loop

When a set of parameters is large, an alternative approach has to be considered. Instead of exhaustively trying every possible combination of parameters, an iterative feedback loop can be applied, as illustrated in Figure 10. In this approach, an initial set of parameter values is selected as a starting point, often based on prior knowledge or reasonable assumptions. The system is then evaluated, and its performance is measured using predefined metrics. The results of this evaluation serve as feedback when adjusting the parameters.

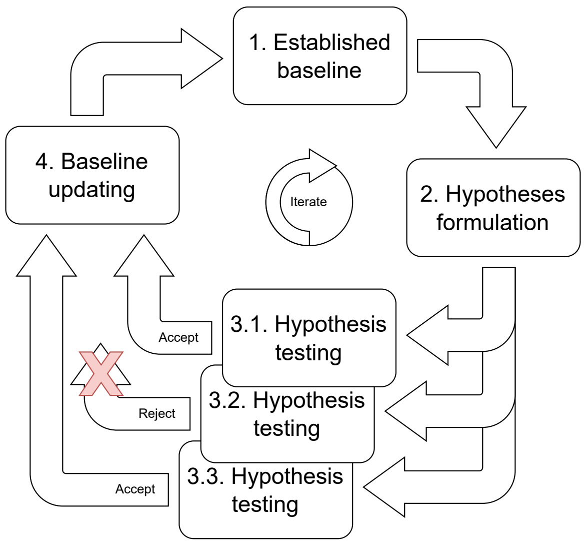


Fig. 10. Iterative feedback loop diagram

This adjustment process is repeated over multiple iterations, with each step refining the parameter values to improve the performance. Compared to the grid search, this approach is more efficient when dealing with a large number of parameters as it avoids evaluating all possible combinations. Since it explores only a subset of the parameter space, the iterative feedback loop method may converge to a local optimum instead of the globally best solution. Nevertheless, it offers a practical trade-off between computational efficiency and solution quality, making it well suited for complex models with many hyper-parameters.

Controlled Randomness

In the field of simulation-based calculations, many decisions are based on a random choice. Some of the examples of employing randomness during previously described methods include the selection of action in the epsilon-greedy algorithm, the selection of random transition from the replay buffer, and the addition of random noise to the calculated route segment boundaries. Throughout the process of hyper-parameter tuning and iterative feedback loop in particular, we base our decisions on the observed results from two simulations with different sets of parameters. However, if the random choices made in the two simulations are entirely different, then arguing about a certain set of parameters being favourable over the other is imprecise.

In order to ensure a fair comparison between the iterations of the hyper-parameter selection procedure, a common approach is to seed the random number generator with a fixed value, thus ensuring reproducibility of the results. However, while seeding the random number generator enables deterministic calculations, it poses the risk of over-fitting the parameters to a particular random number generator seed. In order not to report the final results with an overestimated quality, one must not reuse the same random number generator seed from hyper-parameter tuning procedure in final calculations.

3 Results

This chapter discusses the detailed process of building the components needed for an energy-efficient train control recommender system using reinforcement learning and the results obtained throughout the process. In Section 3.1 we demonstrate how height profile and speed profile are constructed for the target route. Section 3.2 expands on the train movement simulation process with an explained visualisation of a reference trip. The process of arriving at an optimal set of train resistance parameters is described in Section 3.3. The proposed method for route segmentation is described in detail in Section 3.4 explaining the three steps that generate different segment boundary locations. In Section 3.5 we present two formulations of reward functions that achieve the goals of punctual train operation and energy-efficient train operation. From Section 3.6 onward we describe the process of iterative feedback loop being applied to the baseline reinforcement learning implementation, spanning four iterations in Section 3.7, Section 3.8, Section 3.9 and Section 3.10. Finally, a demonstration of results achieved by the optimal model in terms of energy savings is presented in Section 3.11.

3.1 Route Profiles

From the data set containing information about the trip from Kybartai to Paneriai, we visualise the data points on the map. At each location point, we perform a query to Google Elevation API to find the height above the mean sea level for that location. Visualisation in Figure 11 is achieved using Folium[49] library, where the colours at each data point represent the elevation in meters, as indicated in the legend at the top right corner.

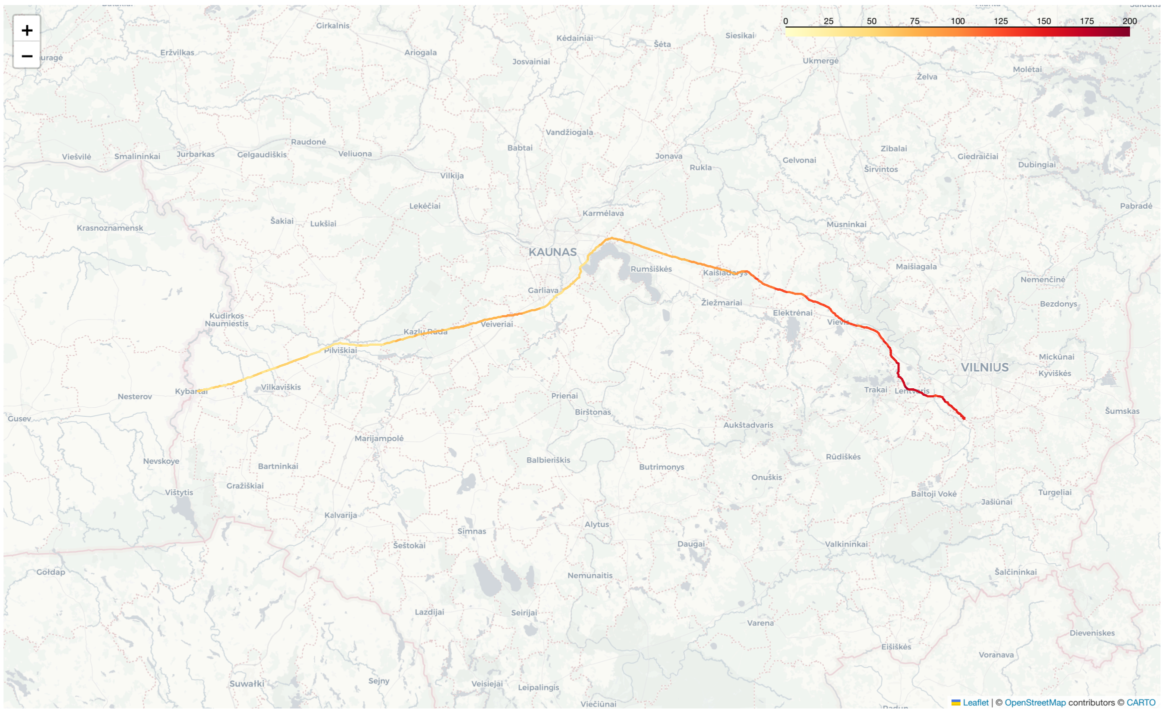


Fig. 11. Route visualisation with overlayed information about track height

Height Profile Smoothing

As discussed in Section 2.1, we perform elevation smoothing using Gaussian kernel, while considering different standard deviation parameter σ values. Figure 12 demonstrates the smoothing effect obtained when σ ranges from $200\ \mathrm{m}$ to $1500\ \mathrm{m}$. We argue that $\sigma = 1000\ \mathrm{m}$ offers a suitable compromise for noise cancellation while relatively closely following the trend observed in the data points.

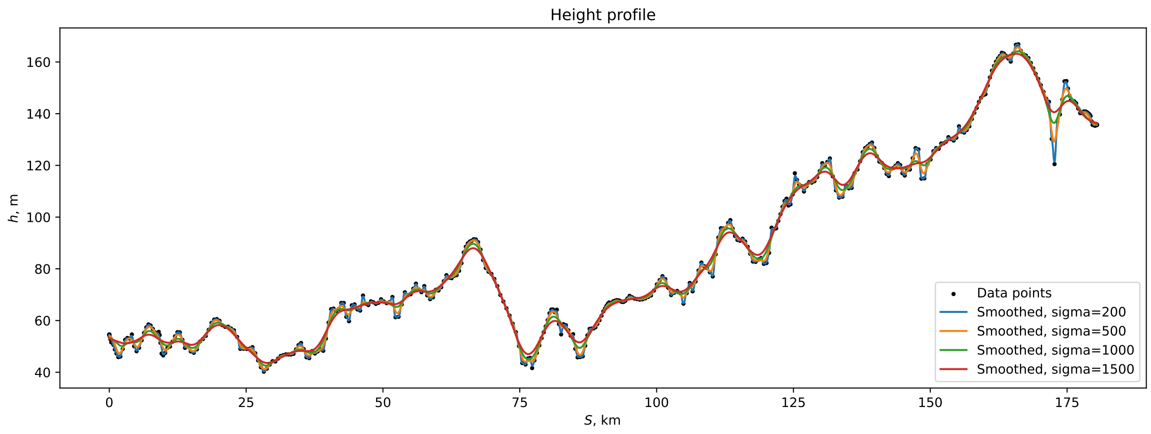


Fig. 12. Height profile smoothing with varying standard deviation parameter σ values

Speed Profile Visualisation

While speed limitations are imposed throughout the entire track, the least restrictive limit is set to $90\ \mathrm{km/h}$ that is applicable whenever no other speed limitation is present. Additional limited speed zones are specified in the data set as a list of distance intervals along the route with a maximum speed value assigned to them. For a better usability, we restructure this information into a function whose value can be accessed at every location along the route and visualise the speed limits in Figure 13.

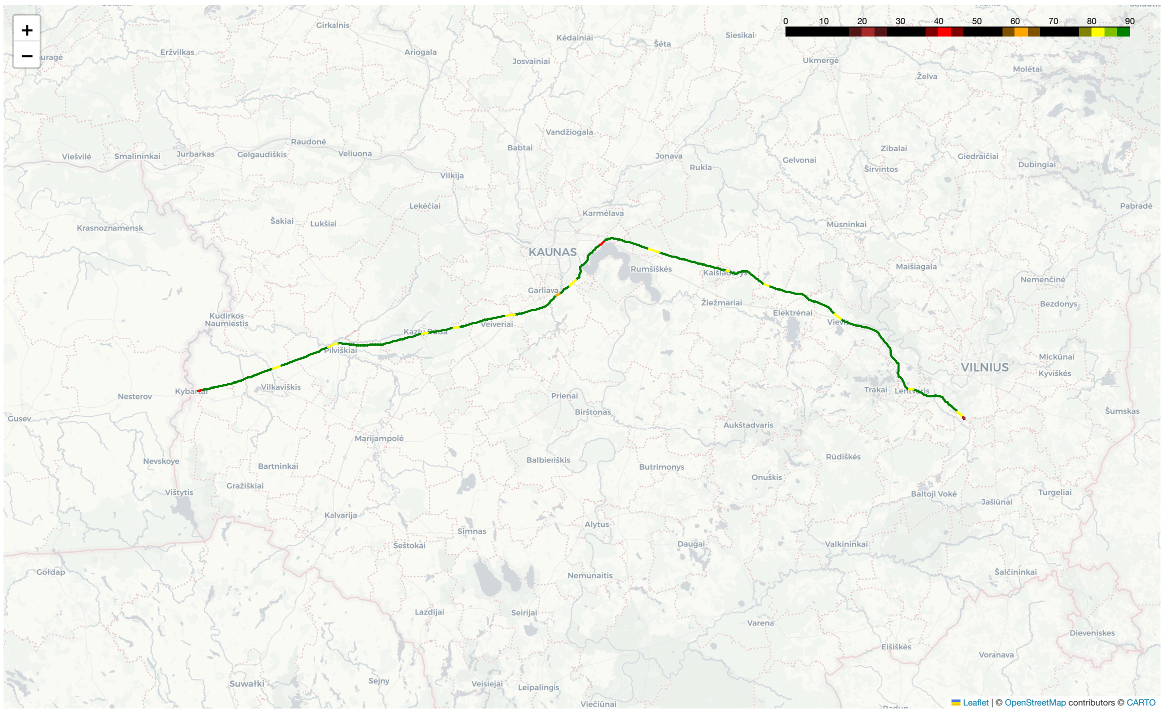


Fig. 13. Route visualisation with overlayed information about speed limits

3.2 Train Movement Simulation

We implement a step-by-step train movement simulation over uneven terrain at a set power level. In this and every upcoming train movement simulation the step size is set to $dS = 50\ \mathrm{m}$, which acts as a reasonable compromise ensuring relatively fast computation times while being sufficiently small for the assumptions about track linearity to hold. At every step dS, the energy balance Equation (10) for the train is continuously formulated and approximated numerically using scientific computing library Scipy[50] to find the train running speed at the end of each such step. When using the set of parameters listed in upcoming Section 3.3, the obtained route profiles are visualised in Figure 14. The blue line corresponds to total time travelled and blue dot at the top right represent the intended time target $t = 15000\,\mathrm{s}$ for the trip, which in this example train simulation gets violated. The green line is the power level at which train engine is running. The red line corresponds to smoothed route height profile. Finally, orange line shows the running speed of the train and orange region corresponds to the speed limits, as visualised in Figure 13. In this simulation, a fixed power level of $800\,\mathrm{kW}$ was selected for the entire trip. We can see how the changes in train speed correlate to the height profile, where the uphill segments are traversed with lower speed values compared to downhill segments. Moreover, note the trimming done to train speed at around $75\,\mathrm{km}$ mark along the route, as explained during the discussion about the operational constraints in Section 2.4.

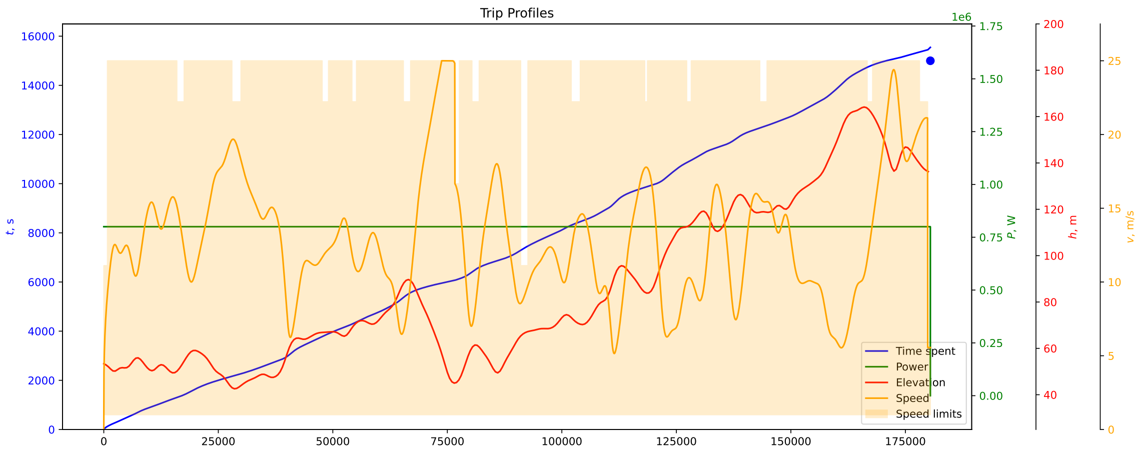


Fig. 14. Train movement simulation outcome

3.3 Resistance Parameter Estimation

While the train movement simulation in Figure 14 can be argued to follow the expected behaviour of physical motion, it was achieved using sub-optimal parameters. Therefore, it represents an arbitrary train moving along the track rather than a specific trip described by the data set. In order to validate that the physical train movement model matches the data sufficiently well, we perform a grid search for parameter values μ and C_d while keeping train descriptor parameters A and m set to fixed values taken from the data. The friction coefficient μ is selected from the range of values between 0.001 and 0.005 in increments of 0.00025 while air drag coefficient C_d is selected to be between 0.1 and 1.5 in increments of 0.05. For every combination of parameter values, we perform a train movement simulation from the real data, as described in Section 2.2. The terminal train speed values are compared at each segment from the data set, excluding those where braking was recorded. The mean squared error metric $\epsilon_{\mathrm{MSE}} = 0.244\,\mathrm{m/s}$ is achieved when using an optimal set of parameters μ and C_d. The differences between the terminal train speed from the simulation and from the real data are illustrated in Figure 15, where the starting velocity of the simulation is reset to match the data at each segment.

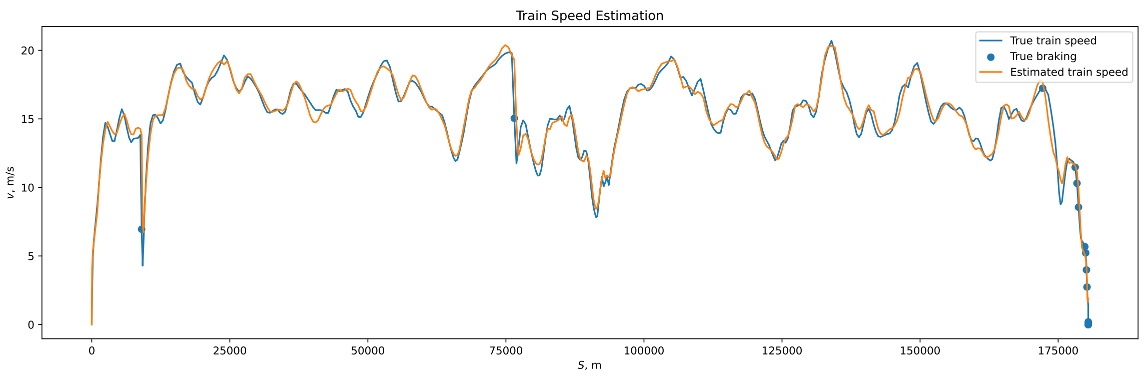


Fig. 15. Train terminal speed comparison between simulation and real data while resetting the errors at each segment

An alternative approach for performing the terminal velocity comparison between simulation and real data could allow the differences to accumulate continuously after each segment. Figure 16 displays the respective mismatches in train movement simulation between the simulated and real data. Note that while the speed profiles have visible differences, especially at locations where the data indicates that braking was performed, they display a significant correlation nonetheless. In this formulation of terminal velocity estimation, the mean squared error term increases up to $\epsilon_{\mathrm{MSE}} = 3.34\,\mathrm{m/s}$.

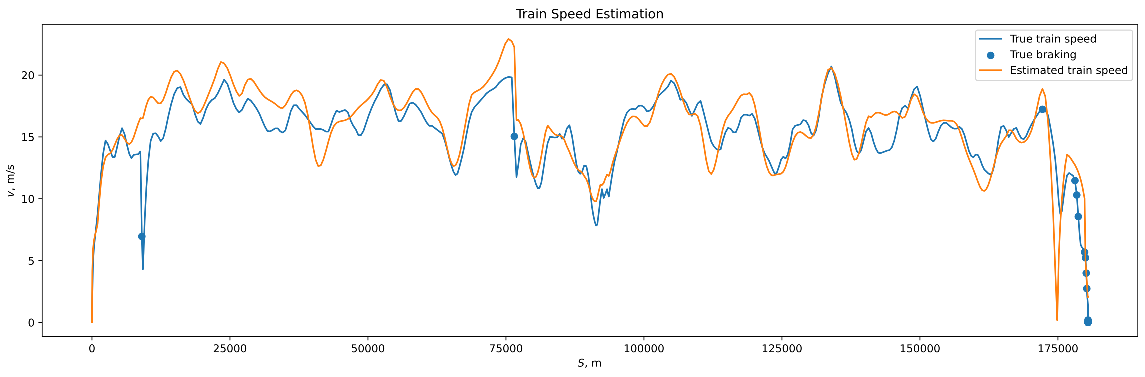


Fig. 16. Train terminal speed comparison between simulation and real data while accumulating the errors

In order not to disclose sensitive information about the company, we refrain from specifying the exact optimal parameter values that were obtained. Instead we set the parameter values to be closely correlated with the estimated values, although not precisely matching. Throughout the remainder of this work and in previously displayed Figure 14, the parameter values used for simulations are thus $\mu = 0.0032$, $C_d = 0.8$, $A = 12.5\,\mathrm{m}^2$ and $m = 1750000\,\mathrm{kg}$.

3.4 Route Segmentation

The method proposed for route segmentation, as explained in Section 2.3, consists of three stages. Initially, the segment boundary points are determined based on the speed limit changes and the slope changes. The segments are then subdivided into smaller ones in order not to exceed a pre-specified maximum length. Finally, a random Gaussian noise is added to the obtained segment boundaries.

Segmentation by Speed Limits

Given that the speed limits are defined as the function of the travelled distance along the route, the route segmentation boundaries can then be expressed as the points where speed limit values change, either by increasing or decreasing. The route segmentation based on speed limits is illustrated in Figure 17, where the blue dots correspond to segment boundaries, as projected on the smoothed height profile displayed in red. In total, 29 segment boundary points are generated from the speed profile.

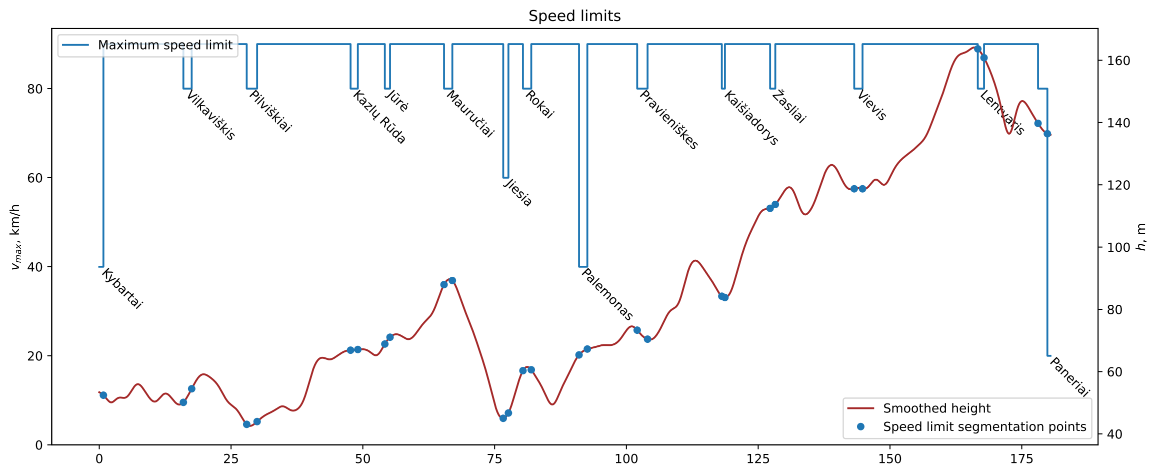


Fig. 17. Route segmentation on maximum speed limits changes

Segmentation by Slope Changes

For route segmentation based on the slope of the track, we propose a method that utilises the insights from the derivative of the route height profile, while allowing for minor fluctuations around the points where the derivative crosses zero, as displayed in Figure 18. Even though the height profile is smoothed using the Gaussian kernel regression, a large number of local optima remain present within the height profile. In order not to segment the route excessively, we define the slope toleration threshold as $h'_{\text{thresh}} = 0.002$. By finding the threshold crossing locations with positive direction $\text{dir}(k) = +1$ and negative directions $\text{dir}(k) = -1$, we discover the points where the route height profile transitions from being increasing to decreasing and vice versa. The segment boundary points are then displayed as green dots on the height profile plot only when the respective slope crossing is followed by a previous crossing in a different direction. In total, 20 segment boundary points are generated using this method from the derivative of the height profile.

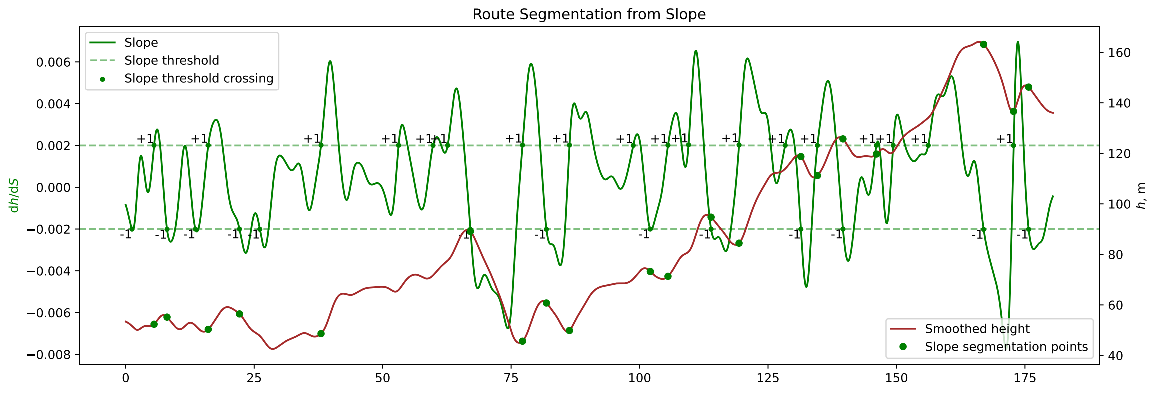


Fig. 18. Route segmentation on slope threshold crossings

Combined Segmentation with Segment Length Refining

The two previously described route segmentation methods, namely, segmentation by speed limits and segmentation by slope changes, are combined together to produce a set of 49 segment boundary points. Since there are a number of segments that are relatively large, we further subdivide each segment with maximum allowed segment length defined as $\Delta S_{\text{max}} = 6000\,\text{m}$. The segments that exceed the length of ΔS_{max} are divided into a minimum number of segments of equal length such that they all do not exceed ΔS_{max}. The combination of segmentation boundary selection choices made by the different stages of route segmentation procedure is displayed in Figure 19, where blue dots represent route segmentation by speed limits, green dots represent route segmentation by slope changes and red dots are the additional segmentation points produced within the segments that remained too long after the other two segmentation steps. In total, 11 additional segmentation points are added, resulting in a total of 60 segmentation points and thus 61 total segments.

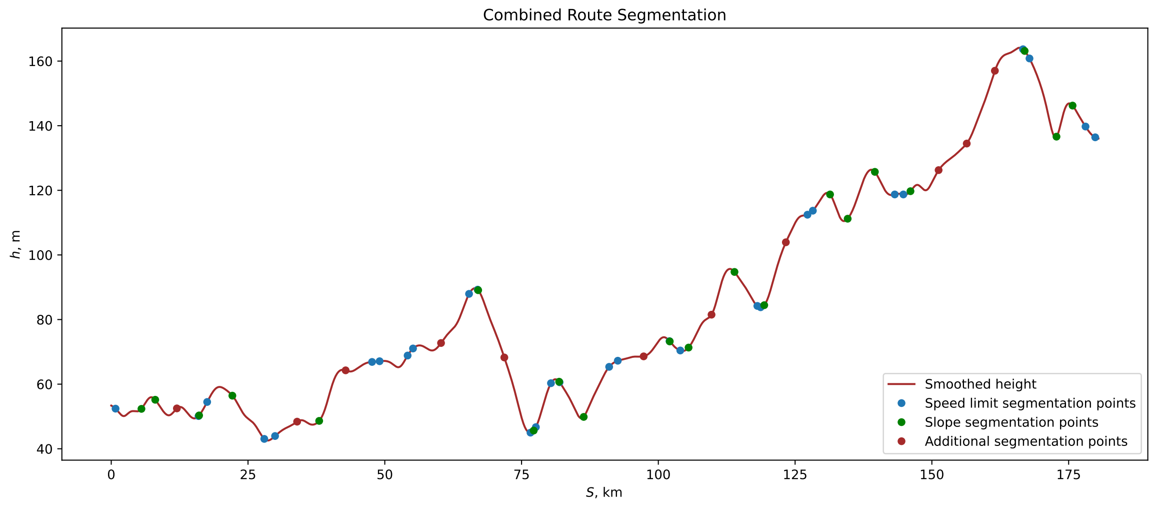


Fig. 19. Combined route segmentation strategy, with an additional step for counteracting long segments

Additionally, each segment boundary is augmented with Gaussian noise $\mathcal{N}(\mu, \sigma)$ that has mean value $\mu = 0.0$ and standard deviation $\sigma = 5.0\,\text{m}$, enhancing the exploration of the learning agent.

3.5 Reward Function Design

For the initial development of the reinforcement learning solution, we mainly rely on punctual train operation reward function, while in further simulations we update the reward function definition to encompass the goal of energy efficiency. We choose to define the reward functions for the two goals using a single structure as in Equation (33).

$$r = c_{\text{frac}} \cdot r_{\text{base}} + c_{\text{frac}} \cdot r_{\text{punish}} + r_{\text{goal}} \tag{33}$$

here, r is the reward assigned after the interaction with the environment, c_{frac} is the coefficient that corresponds to the fraction that the segments constitutes of the entire route, namely, $c_{\text{frac}} = S_{\text{segment}}/S_{\text{total}}$. The components r_{base}, r_{punish} and r_{goal} are the parts of the reward function assigned to the agent for following the base trend, for violating the constraints and for reaching the set goal in time, respectively. For both problem formulations, the punishments and the final reward on successful completion are identical, as expressed in Equation (34) and Equation (35), respectively.

$$r_{\text{punish}} = -0.2 \cdot 1_{\text{not\_on\_schedule}} - 0.4 \cdot 1_{\text{already\_late}} - 0.4 \cdot 1_{\text{speed\_violated}} \tag{34}$$

here, r_{punish} is the component that is responsible for informing the agent about violations, while indicators $1_{\text{not\_on\_schedule}}$, $1_{\text{already\_late}}$, and $1_{\text{speed\_violated}}$ describe possible conditions when the agent receives said punishments. In particular, the punishment of weight 0.2 is assigned when the agent does not follow the intended schedule but is not necessarily late for the time target, and two punishments of weight 0.4 are assigned if the agent is already late for the time target or if its speed drops below a minimum value threshold of $v_{\min} = 1.0 \text{ m/s}$, as defined by a self-imposed minimum speed requirement. Note that maximum speed violations are prevented within the train movement simulation, thus not being included in the reward function formulation.

$$r_{\text{goal}} = +1.0 \cdot 1_{\text{target\_time\_achieved}} \tag{35}$$

here, r_{goal} is a one-time reward for the agent if it reaches the goal within the set time target and $1_{\text{target\_time\_achieved}}$ is the indicator that specifies if the target was reached in time but no earlier than 10% before the intended time. While punishment and final completion reward components are identical, the formulation of the base reward differs for the two definitions of the reward function.

- In case of the punctual train operation, we define the expected speed $v_{\min} \leq v_{\text{expected}} \leq v_{\max}$ at every location along the route in such a way that running the remainder of the trip at this speed would allow the train to reach the target precisely on time or be late as least as possible. The base trend reward is then formulated through the ratio between time spent in the segment and the expected time spent in the segment, as in Equation (36).

  $$r_{\text{base}} = 0.5 \cdot \left(1 - \left|1 - \frac{t_{\text{spent}}}{t_{\text{expected}}}\right|\right) \tag{36}$$

  here, r_{base} is the reward component assigned for following the base trend of punctual operation, t_{spent} is the amount of time spent in current segment and t_{expected} is the expected time spent if the segment was traversed at the expected speed v_{expected}, as defined earlier. While the base reward component is defined so that it cannot exceed 0.5, the lower bound does not exist for this expression. To facilitate a more stable learning process, we truncate the base reward component to $-1 \leq r_{\text{base}}$.

- When deriving the base trend reward component for the energy-efficient train operation, we formulate the theoretical maximum train energy consumption $E_{\max}$ throughout the entire route and use it as a baseline for evaluating the energy savings. We assume that the train moves along the track for the maximum allowed duration $t_{\max}$ at a constant maximum power level $P_{\max}$. In reality, if the train runs at a maximum power level, it is expected to arrive at the destination earlier than the set time target. Therefore, the theoretical maximum energy consumption $E_{\max} = P_{\max} \cdot t_{\max}$ is only hypothetical. For a particular route segment, we consider a fraction from $E_{\max}$ that corresponds to the length of the segment compared with the entire length of the route, namely, $E_{\max,\text{ segment}} = E_{\max} \cdot S_{\text{segment}}/S_{\text{total}}$. The reward function after interaction with the environment is then formulated as in Equation (37).

$$r_{\text{base}} = 1 - \frac{E_{\text{spent}}}{E_{\text{max, segment}}} \tag{37}$$

here, r_{base} is the reward component assigned for following the base trend of energy efficiency, E_{spent} is the energy consumption throughout the segment and $E_{\text{max, segment}}$ is the theoretical maximum energy consumption weighted by the length of the segment. Although the base reward component cannot exceed 1.0 due to E_{spent} always being non-negative, there are practical situations when E_{spent} can exceed $E_{\text{max, segment}}$, potentially resulting in negative reward values. For instance, when the agent chooses a low power level at an uphill section and the speed of the train falls to the minimum allowed value of $v = 0.1\ \text{m/s}$, the duration of the route segment becomes extended and energy consumption formula $E = P \cdot t$ gets dominated by the large value of t even though P is relatively small. Nevertheless, we choose not to truncate the reward function component up to a set minimum because we consider negative values to be an anomaly rather than expectation.

3.6 Reinforcement Learning Baseline Implementation

We implement the reinforcement learning solution to the problem of punctual train operation. Through the method of trial and error, we discover that a suitable set of training loop hyper-parameter values includes $\epsilon_{\text{max}} = 1.0$, $\epsilon_{\text{halflives\_count}} = 8$, $\alpha_{\text{max}} = 0.0001$ and $\alpha_{\text{halflives\_count}} = 2$, where the former pair provides a healthy balance between exploration and exploitation patterns used by the agent throughout the training and the latter pair interacts well with the magnitude of reward function values when updating the parameters of the policy network, while also producing the effect of learning rate decay as the training loop progresses.

Unless otherwise specified, the following parameters will continue to be used as described below throughout the remainder of this work.

- The number of episodes for the training process of the reinforcement learning application is set to 3000.
- The discount factor γ for decaying the upcoming rewards is set to 0.99.
- The replay memory has the capacity of 1000 entries and gets sampled in batches of 128 entries.
- The target network gets updated periodically every 20 episodes by cloning all parameter values from the policy network.
- The policy network regularisation is not configured.

As the initial baseline implementation of reinforcement learning, we simulate the process that is configured for achieving the solution for the problem of punctual train operation while already having the state space regularised, as described in the discussion about the state space transformations in Section 2.4. Additionally, a very simplistic route segmentation strategy is implemented that sets the route segment boundaries at exact multiples of $S_{\text{segment}} = 6000\ \text{m}$ throughout the route, with the last segment being the remainder of length $S_{\text{segment}} = 450\ \text{m}$. The loss function plot and the reward function plot throughout the process can be seen in Figure 20.

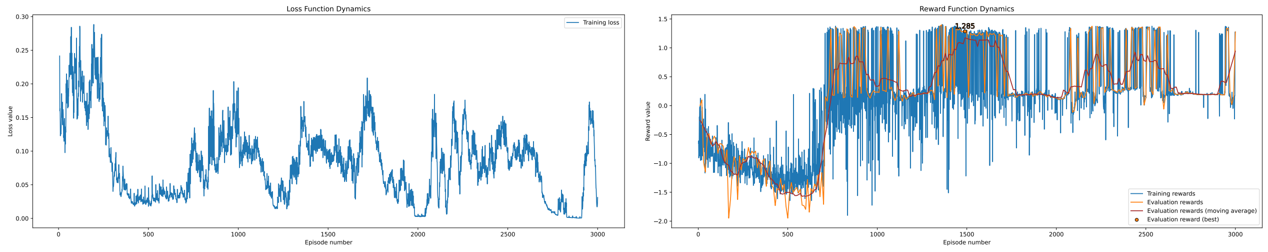


Fig. 20. Loss and reward function plots throughout the training loop of the baseline experiment

We can observe that the convergence of reward function values is not immediate and happens at around episode 750. The training loss plot displays significant fluctuations that sometimes approach values closer to zero, however, this is achieved during episodes when the reward value stabilises around the sub-optimal range of values, which is not a desired condition. Nevertheless, the procedure displays its ability to learn, as indicated by the increasing trend of the reward function values throughout the training loop. Additionally, we display the visualisation of the trip profile in Figure 21 that was produced by the optimal episode with the achieved reward value of 1.285, the selection process of which was discussed in Section 2.5.

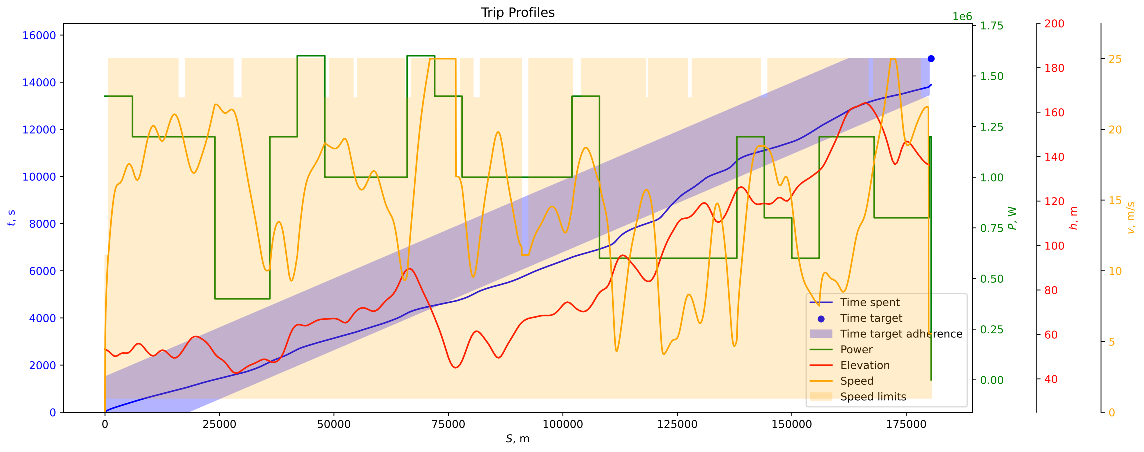


Fig. 21. Optimal trip profile acquired from the baseline experiment

As can be seen in the plot, the train movement simulation mostly follows the blue bound that indicates the intended schedule for the trip, defined as a 10% toleration around the operational strategy that would move at a constant speed and would reach the time target at an exact intended time. In case of the route segments that do not follow the intended bound, namely, between $75\ \text{km}$ and $125\ \text{km}$, a penalty term $-0.2 \cdot 1_{\text{not\_on\_schedule}}$ is assigned, weighted by the total segment length in which the violation was present. Despite the trip profile proposing significant speeding in the first half of the route, the agent managed to strategically slow down by the end in order to not to arrive earlier than 10% before the intended time target.

In the upcoming sections, we describe the creative process for arriving at an improved set of reinforcement learning parameters via the methodology of iterative feedback loop, which help to achieve a more stable model training and the overall higher quality of the solutions. Note that throughout every iteration, the candidate parameters for making adjustments are selected in such a way that results in only minor interactions between them, allowing for a better parallelised experiment execution. Therefore, the practical aspect is preferred during the selection rather than a thematic grouping. After each iteration, a set of adjustments over the baseline implementation, that are considered worthwhile, are included to the next iteration while some of less beneficial changes are rejected. In total, four iterations are presented.

3.7 Iteration: State and Simulation Adjustments

In the first iteration of improvements, we experiment with following parameters.

- The state space regularisation is disabled in order to demonstrate the effects to training stability.
- A portion of random noise is added to the route segmentation boundaries as a way of imitating the imprecise actions made by the operator person.
- A minor modification is made to the route segmentation strategy so that it forms all segments to be of equal length for a more consistent decision making of the agent.
- The additional information about the upcoming route elevation is added to the state representation to demonstrate the improved convergence during learning.

State Regularisation

From the baseline implementation, we exclude the effect of regularisation of the state while keeping the other configuration choices unchanged. In Figure 22 we display the loss and reward function values throughout the training loop, while also showing the respective values obtained from the baseline experiment to act as a reference on the same plot.

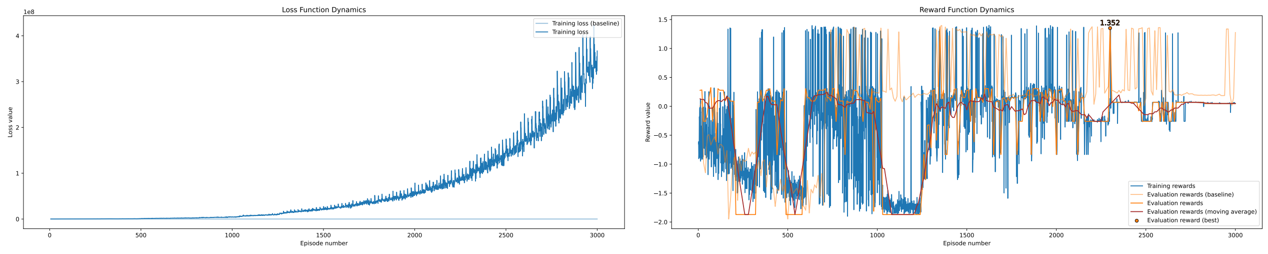


Fig. 22. Loss and reward function plots throughout the experiment with non-regularised state values

The exponential increase of the loss function values undoubtedly provide argumentation that the addition of state space regularisation is crucial for achieving a stable learning process. Moreover, the reward function dynamics also display a deteriorated performance, where the convergence is only achieved at the end of the training loop and the reward values that are approached are far from optimal. The overall optimal solution from the experiment is discerned to have the value of 1.352, which is not a low value in particular, however, the values in this range were only encountered two times during the training loop, suggesting that they are likely produced by accident rather than arguable improvements of the model.

Randomised Segment Boundaries

Given the baseline implementation, we add the ability to simulate the route segmentation boundaries with a random Gaussian noise $\mathcal{N}(\mu, \sigma)$ added to them, having the mean value $\mu = 0.0$ and the standard deviation $\sigma = 5.0\,\mathrm{m}$. In Figure 23 we display the loss and reward function values throughout the training loop, while also showing the reference values obtained from the baseline experiment.

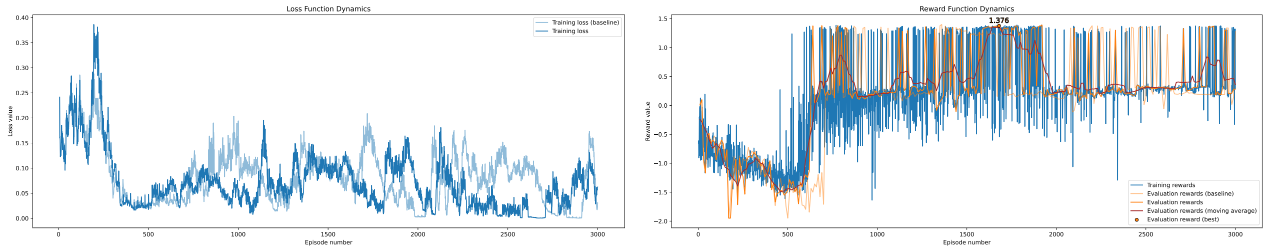


Fig. 23. Loss and reward function plots throughout the experiment with randomised segment boundaries

Even though the initial increase of loss function values seems to have intensified, it then stabilises in the upcoming episodes without showing neither a significant improvement, nor degradation of performance. The same observation also holds for the reward function plot, where even though the convergence to more stable reward values happens slightly faster, the difference is not significant to draw a decisive conclusion.

Nevertheless, we argue that the addition of randomness in the segment boundary definition enables a more realistic simulation of the real-world conditions without degrading the performance of the model, resulting in a better model generalisation.

Similarised Segment Length

The baseline implementation uses the segmentation strategy that generates segments of pre-specified fixed length set to $S_{\text{segment}} = 6000\,\mathrm{m}$. An alternative simplified route segmentation strategy is formulated such that divides the route into the same number of segments, all of which are of equal lengths. Thus, all 31 segments are redefined to be of length of approximately $5821\,\mathrm{m}$. In Figure 24 we display the loss and reward function values throughout the training loop, while also depicting the reference values obtained from the baseline experiment.

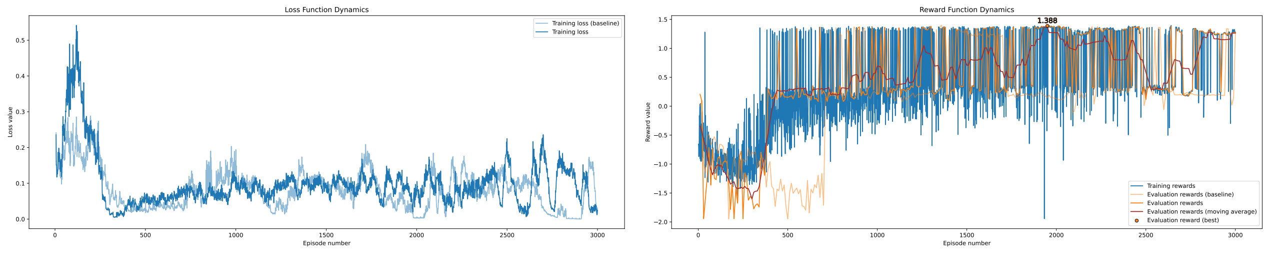


Fig. 24. Loss and reward function plots throughout the experiment with similarised segment lengths

While the loss function plot displays a larger initial increase, its values in later episodes are comparable to the ones received in the baseline implementation. On the contrary, the reward function plot displays a much faster convergence to the close-to-optimal reward function values, suggesting, that by using the segments of similarised lengths we can achieve a faster learning behaviour.

Simple Slope Lookahead

The baseline implementation is expanded with the addition of simple lookahead information about the upcoming track height profile into the state representation. The lookahead information is encoded as the regularised slope from $M = 4$ upcoming segments of length $l = 2000\,\mathrm{m}$, using the notation from Figure 7. The regularisation is performed by dividing the calculated slope value $\mathrm{d}h/\mathrm{d}S$ by a constant factor 0.004, which is estimated to be suitable for the terrain at which the train is moving. The loss and reward function values throughout the training loop are displayed in Figure 25, while also showing the reference values obtained from the baseline experiment.

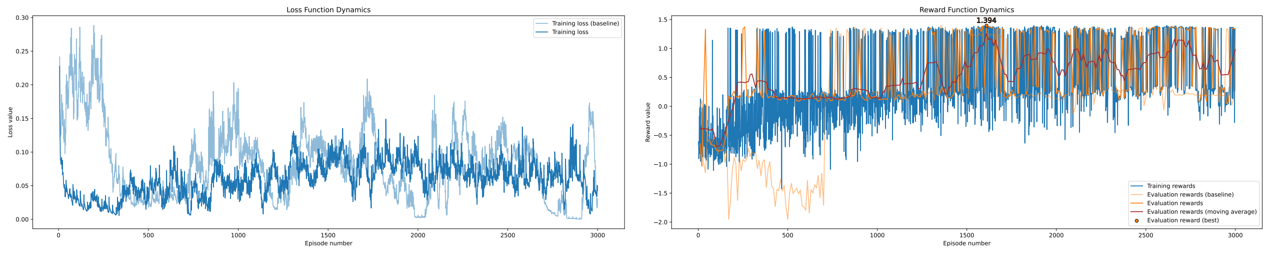


Fig. 25. Loss and reward function plots throughout the experiment with simple slope lookahead encoding

A noticeable improvement to both the loss function plot and the reward function plot is observed after addition of the slope lookahead information into the state representation. In particular, the convergence time of the loss function becomes almost immediate while the reward function values manage to approach the range of close-to-optimal values within the first 200 episodes of the training loop. Additionally, the reward function values at the later episodes of the training loop, although fluctuating between the episodes that manage to achieve the final reward component for adhering to the time target and those that run late, seem to more stably remain around the optimal values. Finally, we observe that the optimal reward value of 1.394 is the highest that we have encountered so far throughout the modifications made to the baseline experiment implementation. We thus conclude that the addition of slope lookahead information can indeed improve the training performance in terms of the faster convergence of the loss and reward function values as well as the overall better quality of the rewards obtained.

3.8 Iteration: Energy Efficiency Reward Adoption

We combine the properties that we consider beneficial from the first iteration of experimentation. In particular, we keep the state representation regularised and alter the definition of the segmentation strategy to produce the segments of the same length throughout the track. Additionally, we configure the added randomness to the route segmentation boundaries and encode additional information about the regularised track slope lookahead into the state representation.

Due to the fact that simple slope lookahead into $M = 4$ segments of length $l = 2000\,\mathrm{m}$ was designed having in mind the initial route segmentation strategy that divided the route into fixed $6000\,\mathrm{m}$ sized segments, now that we transition to a route segmentation strategy with a similarised segment lengths, we analogously update the state representation to instead include four subsegments that each are one third of the size of the segment ahead, as illustrated in Figure 8. The added slope lookahead values $\mathrm{d}h/\mathrm{d}S$ are also regularised using the divisor 0.004.

In Figure 26 we display the loss and reward function dynamics of the experiment that combines the mentioned configuration adjustments from previous iteration. Additionally, we display the same metrics from the baseline implementation of the first iteration to illustrate the combined effect that was achieved using the mentioned adjustments.

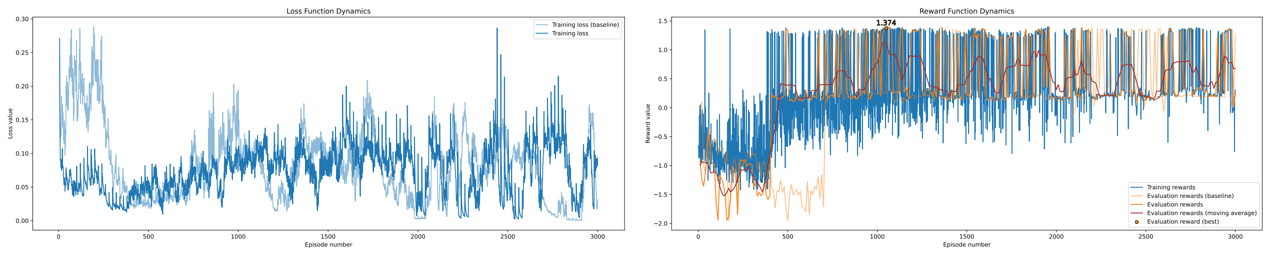


Fig. 26. Loss and reward function plots throughout the experiment that combines choices made in the first iteration

We observe the expected low loss function values and fast convergence that were inherited from the lookahead strategy application in previous iteration. Moreover, while no degradation in performance is apparent in both plots, we managed to expand the problem formulation to account for imprecise actions made by the operator person by including randomness to the segment boundaries as well as rephrased the route segmentation strategy to consider similarised segment lengths. We display the route generated using the train control strategy from the episode with the optimal reward of 1.374 in Figure 27.

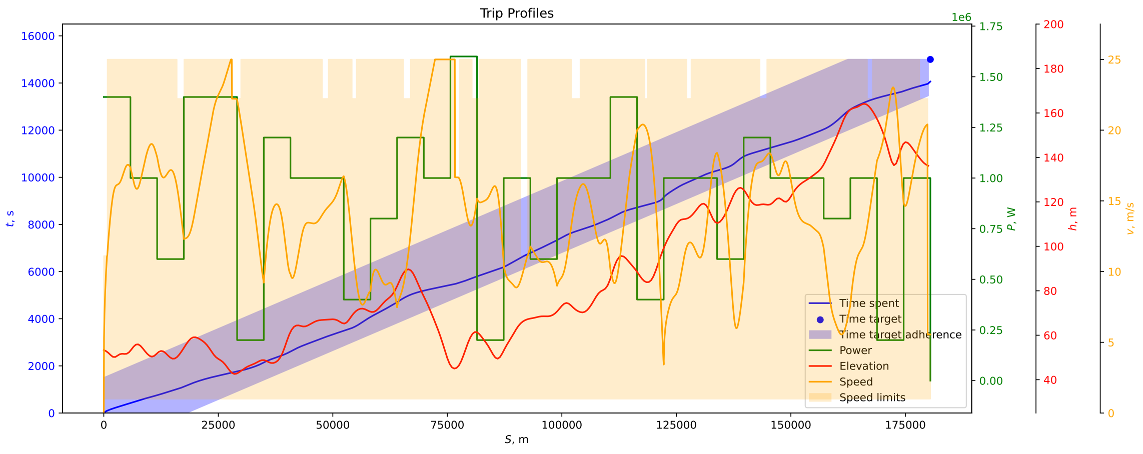


Fig. 27. Optimal trip profile acquired from the experiment that combines choices made in the first iteration

A noticeable improvement can be observed compared the baseline implementation in Figure 21, where the intended time bounds were violated for the segment of around 50 km in length at the middle of the route. In the trip profile acquired from the optimal episode of the latest experiment, the time target is adhered throughout the entire route and the train manages to reach the set time target successfully.

For the upcoming comparison, we redefine the baseline to be the current experiment. The energy efficiency reward function is formulated in the next iterative step and compared with the said baseline.

Energy Efficiency Reward Function

As defined in Section 3.5, we implement the reward function corresponding to the problem of energy-efficient train operation while keeping all other configuration choices unchanged. The resulting plots for loss and reward value dynamics are displayed in Figure 28.

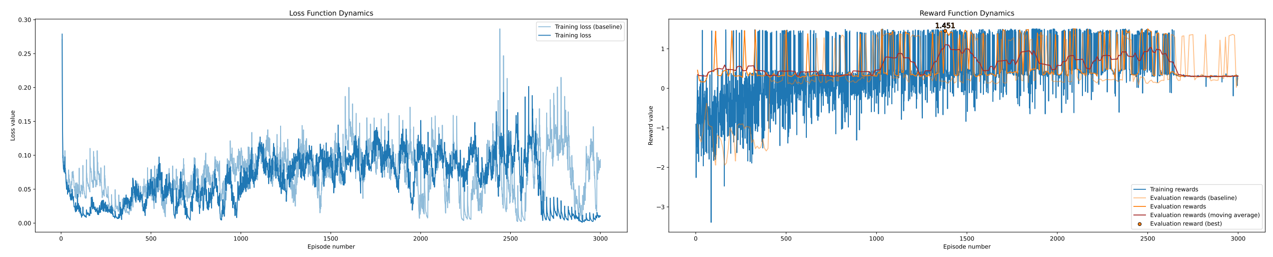


Fig. 28. Loss and reward function plots throughout the experiment after adoption of the energy efficiency reward function

While the loss function value fluctuations are very similar to those in the baseline implementation, a noticeable improvement to the reward function behaviour is observed after the adoption of energy-efficient train control reward function. The convergence to the close-to-optimal reward values is almost immediate, although the model fails to encounter higher value rewards at the end of the training loop. Nevertheless, the optimal episode achieves the reward value of 1.451 with its driving patters displayed in Figure 29.

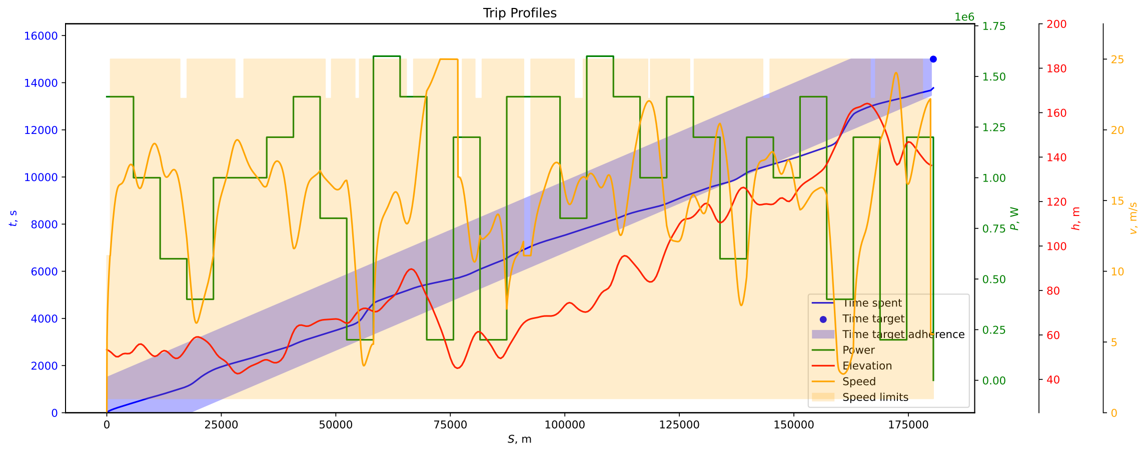


Fig. 29. Optimal trip profile acquired from the experiment after adoption of the energy efficiency reward function

In the the optimal trip profile describing the energy-efficient train movement strategy, we observe that the agent learns to arrive at the set time target while sacrificing a portion of the reward by not precisely adhering to the set time target bounds. More importantly, the expected patterns of energy-efficient driving can be observed in the behaviour of the agent, where lower power levels are set during downhill section while higher power levels are used when moving uphill. Although the reward function definition is updated from the previous formulation for achieving punctual train control to an alternative formulation for achieving energy-efficient train control, the model successfully adapts to the modified task. Since the reward function specifics are different, we cannot compare the reward values directly. However, we observe that the learning behaviour, as displayed in the loss and reward function plots, remains similar when using both formulation.

The adoption of the energy efficiency reward function concludes the second iteration of the system configuration using iterative feedback loop. For further comparisons, we redefine the baseline to corresponds to the current experiment.

3.9 Iteration: Simulation and Neural Network Design Adjustments

In the third iteration of improvements, we experiment with following parameters.

- A randomised train mass simulation is implemented, this way expanding the scope of the problem to make it better generalisable for trips with different cargo loads.
- The route segmentation technique using only changes to slope is implemented, which should allow the agent to make decisions that are better correlated with the changes in terrain.
- An attempt is made to enable dropout regularisation with expectation for it to improve the generalisation properties of the model.
- An alternative approach for softly updating the parameters of the target network is examined, where parameter values are adjusted in a more frequent schedule but only by a fraction of values from the policy network parameters.

Randomised Mass Simulation

In order to better generalise the network to work correctly with different cargo loads, the information about the train mass is included in the state representation, additionally regularised by dividing its value by the expected train mass m from the configuration. Since this information is supplied to the neural network, the training process has to be adjusted accordingly to simulate the fluctuations of train mass during different episodes. The train mass randomisation is defined through the normal distribution $\mathcal{N}(\mu, \sigma)$, where the mean value $\mu = m$ is the mass of the train and standard deviation $\sigma = \sigma_m \cdot m$ is defined through the parameter σ_m that captures the multiplicative effect of the standard deviation, effectively resulting in the train mass being expressed as $m_{\text{rand}} \sim m \cdot \mathcal{N}(1.0, \sigma_m)$.

Although the model training is performed using randomised mass during each episode, the evaluation runs that happen every 10 episodes have to simulate a fixed train mass m in order to generate comparable train energy consumption metrics. While keeping the same configuration as the baseline experiment, with energy-efficient reward function formulation and a single addition of randomised mass simulation, we generate the loss and reward function plots in Figure 30.

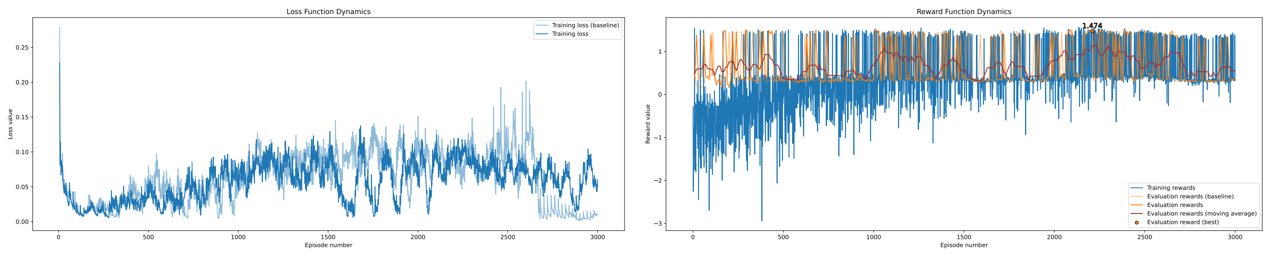


Fig. 30. Loss and reward function plots throughout the experiment with randomised masses being simulated

The convergence properties of the loss and reward functions do not seem to be affected by the addition of randomised mass to the state representation, when compared to the baseline implementation. Nevertheless, the generalisation with the randomised train mass manages to arrive at a better optimal solution with the reward value of 1.474, while producing a network that can be applied in a problem with a broader scope, compared to the baseline that is optimised for a fixed mass only.

Segmentation on Track Slope Changes

We perform route segmentation using slope changes only, as discussed in Section 3.4. While the number of segments into which the route gets divided reduces from 31 in case of simple similarised segmentation approach down to 21 in the approach guided by the slope changes, we choose not to make any adjustments to other parameter values. Although the number of segments present along the route affects the amount of training steps that are performed during each episode, we argue that the difference can be overlooked at this stage of iterative improvement. Namely, the parameter γ that represents the reward discounting on future steps now exhibits a stronger effect when the number of steps is lower. Additionally, the replay buffer parameters, such as batch size and buffer length, that define the duration of the memory, now have the effect of remembering a longer history in terms of previous episodes. The loss function plot and the reward function plot generated by the updated experiment configuration are displayed in Figure 31.

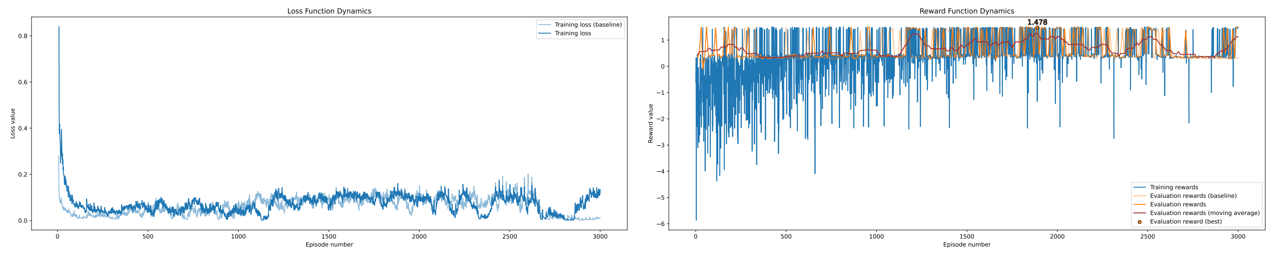


Fig. 31. Loss and reward function plots throughout the experiment with route segmentation based on track slope

The loss function indicates a slower convergence rate, which could be caused by the fact that replay memory takes a longer time to initialize with enough values for training. Nevertheless, further trend of the reward function remains similar to the baseline implementation. Although no apparent improvement is observed in the loss and reward function plots, from the theoretical understanding of the problem, we expect advanced segmentation strategies to better accommodate to the decision making process of the agent. The lack of improvement may be argued by the limitations of the route segmentation strategy implementation, where only a one stage of segmentation is performed, without the following stage of subdividing the segments into smaller subsegments for a better granularity during the decision making process.

Dropout Regularisation

In the neural networks design we add dropout regularisation layers with value $p = 0.5$ indicating that half of parameter weights should be disabled during every iteration of the training loop. This adjustment is made to the policy network only, while keeping the target network unaffected. The respective loss and reward function plots are displayed in Figure 32.

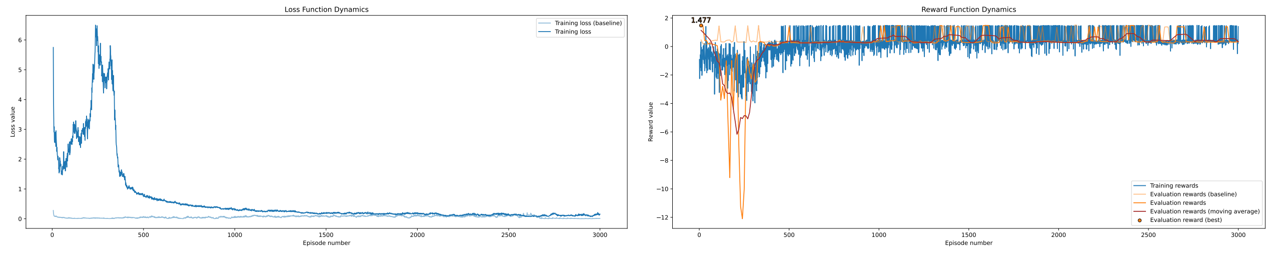


Fig. 32. Loss and reward function plots throughout the experiment with dropout regularisation enabled

The effect from the application of dropout regularisation is observed to be negative in both plots corresponding to the loss values and reward function values. While the loss function does not manage to reach a similar level as the baseline, the reward function exhibits the ability to approach the close-to-optimal reward values. However, the optimal episode for the process is selected to be the first evaluation run ever performed, indicating that the procedure degraded over time and was outperformed by an arbitrary iteration when the state space was yet to be well explored. Instead of selecting this iteration for illustration, we take the model from the last episode and display its proposed trip profile in the Figure 33.

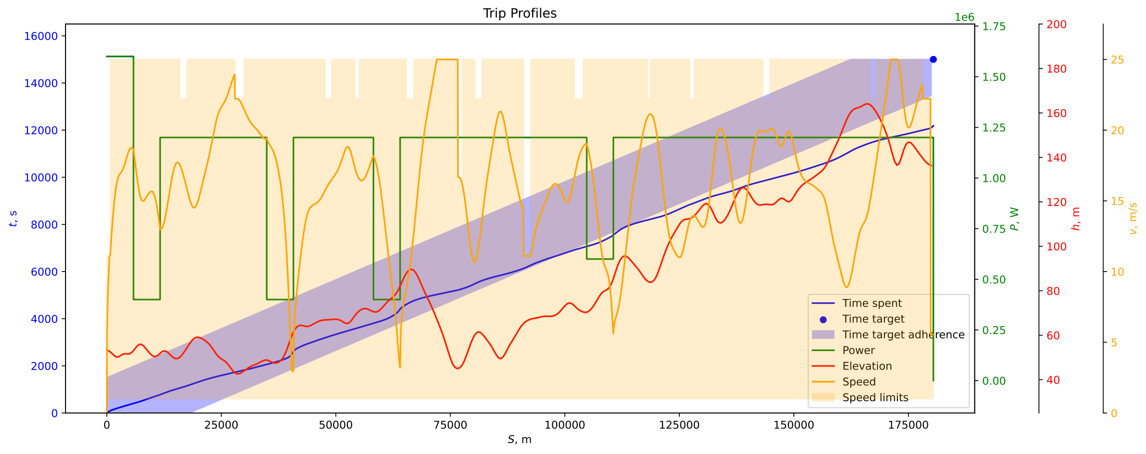


Fig. 33. Overly generalised trip profile acquired from the experiment with dropout regularisation enabled

We observe that the model very strongly generalises the decisions of the agent and practically always proposes a single power level. Moreover, when a different power level is selected, it usually does not match the intended behaviour for energy-efficient train control, where proposals to reduce the power level are made during uphill segments. We thus argue that the addition of dropout regularisation performs poorly in the context of train movement simulation using deep reinforcement learning.

Target Network Soft Updates

Instead of periodically cloning all of the policy network value to the target network every 20 episodes, an alternative formulation of performing soft updating is considered. As defined in the Equation (23), the parameter τ controls the fraction of the parameter values that get updated from the policy network to the target network after the model training is performed in each learning step. In order to make a valid comparison, we select an update rate $\tau = 0.0016 \approx 1 \, / \, (20 \cdot 31)$, which has the same expected value of target network being updated over 20 episodes for 31 steps each, as is the number of segments produced by the similarised route segmentation strategy. The loss and reward functions throughout the training loop are displayed in Figure 34.

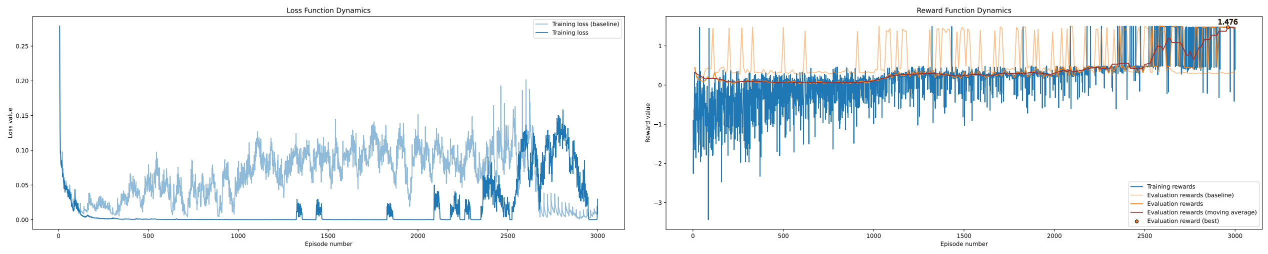


Fig. 34. Loss and reward function plots throughout the experiment with soft updates to target network

Even though the loss value plot indicates a faster and more stable convergence compared to the baseline, the practical inspection into the reward function plot suggests that the model experiences difficulties when aiming to reach higher values of the reward function. However, by the end of the training loop the obtained reward values significantly increase, as also seen in the loss function plot where higher values start to be reported. While it is difficult to argue if soft parameter updating results in a better learning process, we choose to continue the experimentation using periodic updates every 20 episodes since it produces the results that are more naturally interpretable.

3.10 Iteration: Advanced Route Segmentation Adoption

We combine what was learned in third iteration to configure the experiment with the following parameter values to demonstrate that a sufficiently high performance is achieved when using those properties in combination. The concepts of dropout regularisation and soft updating of the target network are rejected from the third iteration of the feedback loop. The adjustments that we include in further experimentation are the randomised mass selection and the segmentation strategy based on the track slope changes. The results of the experiment that contains both these properties as well as the configuration from the first two iterations are displayed in Figure 35 in terms of loss and reward function dynamics. For completeness, we summarise that the experiment contains the following properties.

- State regularisation;
- Fractional slope lookahead encoding into state representation;
- Randomised mass simulation and mass encoding into state representation;
- Segmentation strategy based on track slope changes only;
- Randomised segment boundaries;
- Energy-efficient reward function formulation.

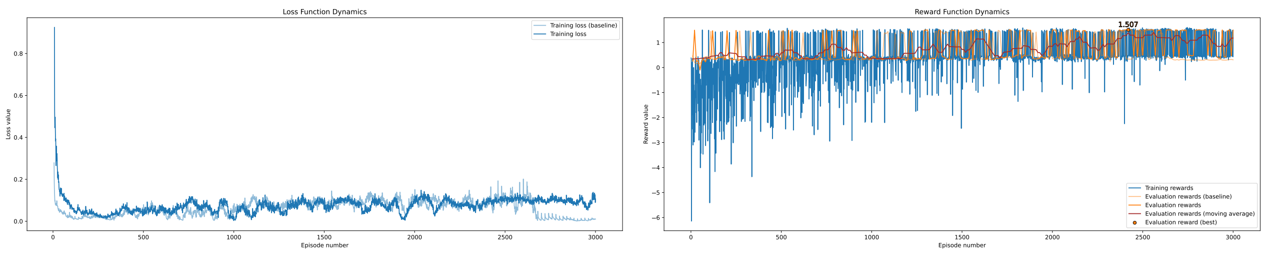


Fig. 35. Loss and reward function plots throughout the experiment that combines choices made in the third iteration

While combining a set of improvements to the performance of the model with another set of attempts to expand the application scope of the model, the plots display the properties of stable learning and reasonably good rewards being generated. This suggests that individual parameters selected throughout the iterative feedback loop work well when applied in combination. The trip profile generated during the optimal episode is displayed in Figure 36.

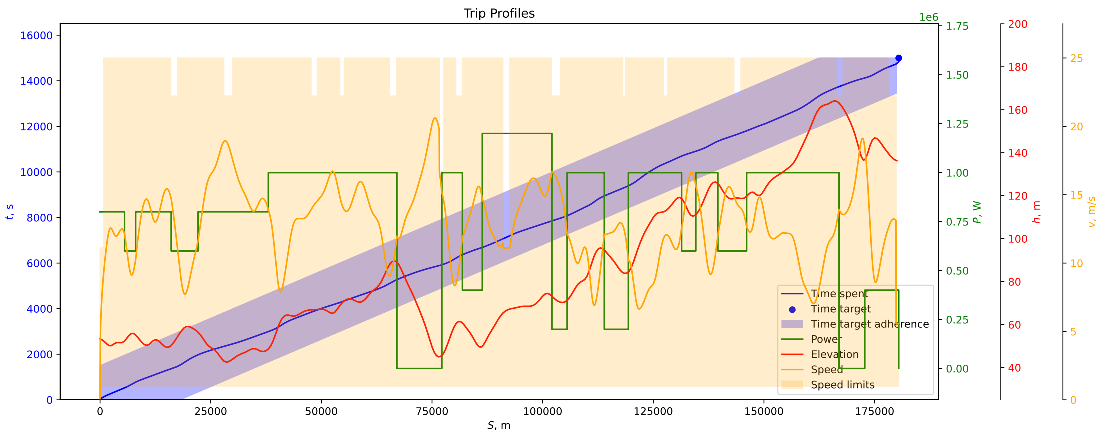


Fig. 36. Optimal trip profile acquired from the experiment that combines choices made in the third iteration

We observe perfect adherence to the set schedule as well as intuitively arguable power level selections, such as turning off the engine entirely in longer downhill segments. Throughout the upcoming iteration, we will refer to the results of this experiment as the new baseline.

Three-Stage Route Segmentation

Finally, we improve the route segmentation strategy to include additional information about speed limit zones and subdivide longer route segments into smaller ones, along with relevant adjustments made to other experiment parameters to accommodate for this transition.

Notably, since we include the speed limit changes as the boundaries for route segments, we also need to convey the relevant information about the enforced speed limits in the state representation. To account for that, we expand the state representation to describe two upcoming route segments in terms of their speed limits, their slope and their length. Given that the number of dimensions in the state representation keeps increasing, we opt to reject the method of dividing the segment into thirds, as proposed previously. Instead, we augment the state with information about two full segments, thus expanding the original regularised state (S, v, t, m) with dimensions $(v_{1\,\max}, h'_1, \Delta S_1, v_{2\,\max}, h'_2, \Delta S_2)$, where $v_{i,\,\max}$ corresponds to the speed limit at the upcoming segment i, h'_i represents its slope and ΔS_i is the length of the segment, all of which are regularised by the expected maximum values.

Additionally, as the number of segments, which initially was 31 and then got reduced to 21, now increases to 61, we choose to adjust the reward discount parameter γ to slow down the decay of future rewards from 0.99 to 0.995. Additionally, we double the capacity of replay buffer to 2000 and training batch size to 256 so that they represent a comparable count of previous episodes. The loss and reward function dynamics can be seen in Figure 37.

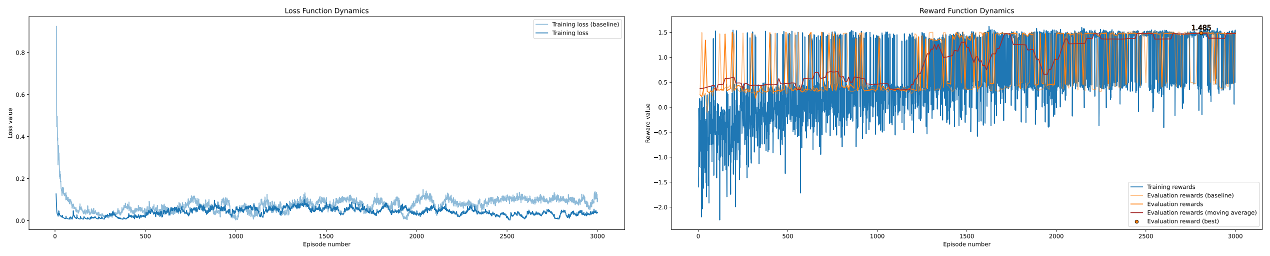


Fig. 37. Loss and reward function plots throughout the experiment with an advanced three-stage route segmentation strategy

Although the loss values do not display a particular improvement when compared to the baseline experiment, the reward graph shows a faster and more decisive convergence to the optimal reward values. Moreover, throughout the training process, the agent does not acquire as low reward values during training as in previous experiments, which suggests a more stable learning process.

Additionally, in Figure 38 we illustrate the trip profile generated during the optimal episode, which achieved the reward value of 1.485. Although it contains more complicated transitions between the power levels, it also displays the intuitive patterns of energy-efficient train operation.

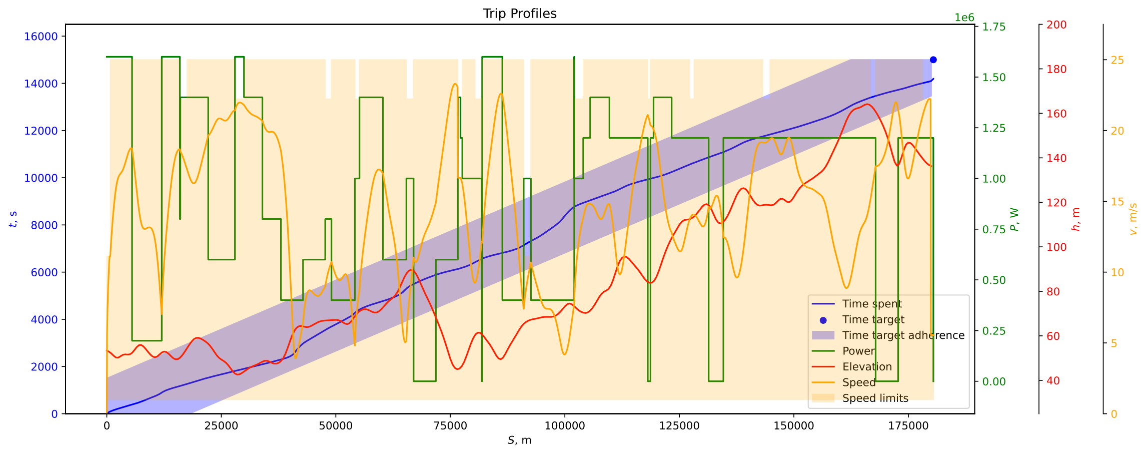


Fig. 38. Optimal trip profile acquired from the experiment with advanced route segmentation

While the expansion of the model to perform in a broader application scope allows for it to be better generalisable, the learning process in general tends to not reach the optimal solution. To account for the expansion of the problem scope, we retrain the model over 5000 episodes rather than the standard number of 3000 used throughout the work. The resulting loss and reward functions are displayed in Figure 39 where the baseline is defined as the same experiment that was run for 3000 episodes.

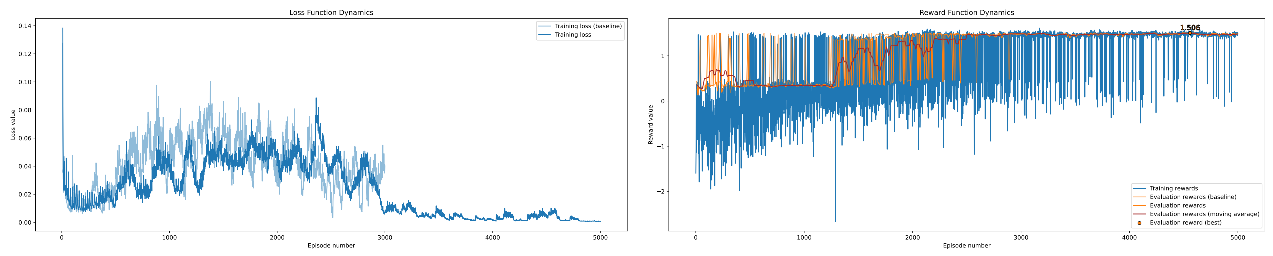


Fig. 39. Loss and reward function plots throughout the final experiment after training for 5000 episodes

A significant improvement in loss function values and reward function stability is observed with the optimal trip profile reaching the reward value of 1.506. The trip profile of an optimal episode is displayed in Figure 40.

Compared to the results obtained from training the model for 3000 episodes, we observe that this route profile displays more edge-case behaviour, that allows to optimise its performance in exchange for the risk of violating the requirements. As the training loop increases in length, the model gains more confidence in its behavioural patters, thus being able to better exploit its understanding of the problem.

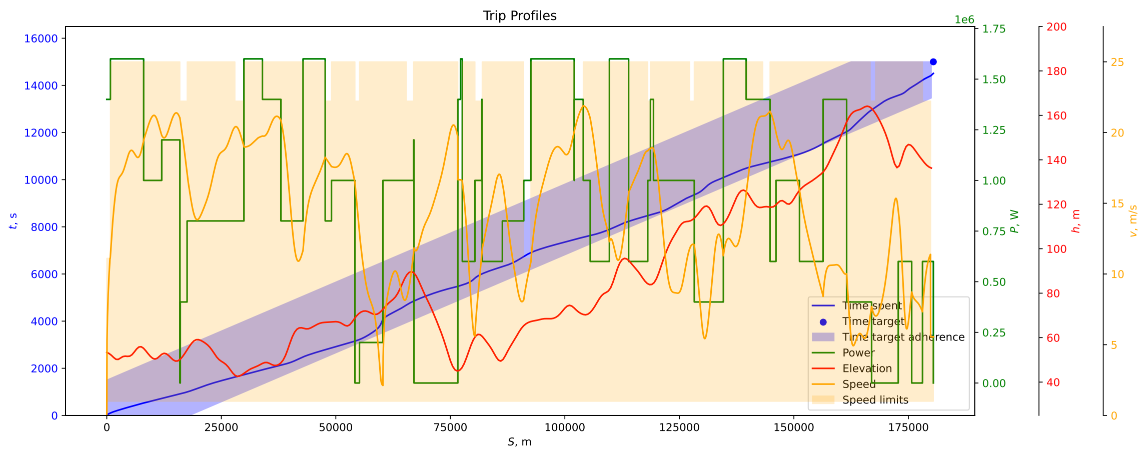


Fig. 40. Optimal trip profile acquired from the final experiment after training for 5000 episodes

3.11 Optimal Model Performance Evaluation

Given that the optimal model in Figure 40 receives the reward value of 1.506 and is not assigned with penalties throughout the simulation, we can deduce that its energy consumption reduction corresponds to 50.6% from the theoretical maximum consumption defined in the reward function description. While such metric does not convey much insights into real-world energy improvement, it still provides a perspective for interpretation of the results that are generated by the algorithm.

A more relatable metric defines the train energy consumption savings against one of the possible trip profiles. For instance, when the train moves the entire route at a constant power level of $800\,\mathrm{kW}$, as shown in Figure 14, it fails to reach the time target by $540\,\mathrm{s}$, which relates to 3.6% of the allowed trip time of $t_{\mathrm{total}} = 15000\,\mathrm{s}$. Throughout the trip, it consumes $1.236 \cdot 10^{10}\ \mathrm{J} = 3453.4\,\mathrm{kWh}$ of energy.

In comparison, the trip proposed by the optimal model takes $14509\ \mathrm{s}$ to arrive at the target location, which is 3.3% shorter than the allowed time target, and consumes a total of $1.184 \cdot 10^{10}\mathrm{J} = 3290.2\mathrm{kWh}$ of energy. When compared to a simple strategy of setting a constant power level, a relative power consumption improvement of 4.7% is achieved. Moreover, it is worth emphasising that the reference trip fails to achieve the time target. In contrast, the proposed method for train movement simulation pays close attention to adherence to the time target while still resulting in a reduced power consumption.

4 Limitations and Recommendations

Throughout the work, we formulated a series of simplifications and approximations that allowed computationally feasible simulation of energy-efficient train control while representing the problem relatively accurately. The potential for improvement is vast, starting from a more physically accurate train movement simulation up to practical considerations for the parameterisation of model training process. In this section, we list a selection of areas where improvement can be made in future work.

1. As we relied solely on the train movement simulation that is based on defined physical models, we may have overlooked many practicalities of applying the trained models in real-world train operation. The evaluation performed in this work used two hypothetical scenarios as benchmarks. Nevertheless, we acknowledge that a consultation with an expert train operator and an opportunity to apply the proposed energy-efficient train control strategy on a real train could lead to better insights on how to improve the training process, either via adjustments to the physical model, the reward function design, or any other aspect that was not considered.

2. While we firmly believe that following a precise route height profile during the physical simulation of train movement can drastically reduce the mismatch between the simulation and real-world train operation, in this work we did not undertake extensive efforts to improve the quality of data that we were supplied with. Instead, we applied data preprocessing, such as height profile smoothing, in order to simplify the terrain over which train movement is simulated. In practice, measuring the true elevation at which the train tracks are laid can be achieved in a single careful traversal of the route using precise tooling. This could reduce the need for data preprocessing steps and result in a more accurate train movement simulation.

3. Throughout the training process of our neural network model, we opted to over-fit for a single route. Although it is a valid and often applied approach for solving problems in the field of reinforcement learning, one can argue that retraining the model for every possible route with every possible set of operational constraints is wasteful. Notably, in our work an attempt was made to include a selection of dimensions, such as randomised train mass and route slope lookahead, into the state representation in order to enable the model to make better informed decisions. Nevertheless, such state augmentation is not sufficient if aiming to produce a fully generalised model that could work on any unseen track profile.

4. When defining the configuration for the reinforcement learning training loop, we simplified the discussion by proposing pairwise choices between experiment conditions with comparable differences in behaviour, as a method of iterative feedback loop parameter estimation. The selection of an optimal set of hyper-parameters is a common problem in many computational tasks, with some of the widely applied methods for solving it being grid search or random sampling from a distribution of hyper-parameters, both of which increase the training duration drastically. However, since our neural network models already took an excessive amount of time to train, we avoided such experimentation due to time restrictions. Assuming that future work in the field is capable of proposing a single model that successfully generalises to any route profile and any set of operational constraints, the application of a sophisticated hyper-parameter tuning methodology could be worth the additional time spent during network training.

Conclusions

1. By formulating the physical model through the law of conservation of energy, we managed to implement a simulation for train movement along the route, which we examined to be representative of the expected behaviour of physical motion. When simulating the train movement with conditions updated from the data at each step and estimated optimal resistance parameter values, the mean squared error of $0.244 \mathrm{~m/s}$ was achieved between estimated and actual terminal velocities after each segment.

2. We have implemented a reinforcement learning solution to the problem of punctual train operation that demonstrated the ability of the agent to learn from the interactions with the environment when aiming to reach the destination within the set time target. The results were motivated by the loss function values approaching zero and the punctual train operation reward function reaching the value of 1.285 during the baseline implementation.

3. Additionally, we expanded the train state representation to include information about the height profile lookahead, thus enhancing the rate of convergence to the close-to-optimal solutions. The improvement from around 750 episodes to 200 episodes needed for convergence is seen in the reward function plot after including height profile lookahead encoding. The optimal reward function value for punctual train operation also increased to 1.394.

4. We expanded the design of the reward function to account for the goal of energy efficiency, which in turn enabled the learning agent to adapt to a new goal and demonstrate the behaviour of energy-efficient train operation. The behaviour of immediate convergence was observed with the optimal reward function value of 1.451 being reported for the problem of energy-efficient train operation.

5. By generalising the problem to allow movement simulation of trains with varying masses, we expanded the scope in which the proposed solution can be applied without affecting the performance of the model. On the contrary, the property of immediate convergence to close-to-optimal reward values was retained and the optimal reward value for the problem of energy-efficient train operation increased to 1.474.

6. To better accommodate to the decision making procedure of the learning agent, we proposed a route segmentation strategy that captures information about significant changes to route elevation in combination with information about the speed limits along the track. With an improved route segmentation approach, the model demonstrated a more stable distribution of reward values during training and achieved the optimal reward value of 1.506 in the context of energy-efficient train operation.

7. Finally, we demonstrated how the train control strategy derived from the optimised deep reinforcement learning solution is capable of generating a solution to the problem of optimal train power control, achieving overall energy savings of up to 4.7% when compared to a simplified baseline train control strategy.

References

[1] Gerben M. Scheepmaker, Rob M.P. Goverde, and Leo G. Kroon. “Review of energy-efficient train control and timetabling”. In: *European Journal of Operational Research* 257.2 (2017), pp. 355–376. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2016.09.044. URL: https://www.sciencedirect.com/science/article/pii/S0377221716307962.

[2] L. Bittner. “L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, E. F. Mishechenko, The Mathematical Theory of Optimal Processes. VIII + 360 S. New York/London 1962. John Wiley & Sons. Preis 90/–”. In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 43.10-11 (1963), pp. 514–515. DOI: https://doi.org/10.1002/zamm.19630431023. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19630431023. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19630431023.

[3] Kunihiko Ichikawa. “Application of Optimization Theory for Bounded State Variable Problems to the Operation of Train”. In: *Bulletin of JSME* 11.47 (1968), pp. 857–865. DOI: 10.1299/jsme1958.11.857.

[4] Gerben M. Scheepmaker and Rob M.P. Goverde. “Energy-efficient train control using nonlinear bounded regenerative braking”. In: *Transportation Research Part C: Emerging Technologies* 121 (2020), p. 102852. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.102852. URL: https://www.sciencedirect.com/science/article/pii/S0968090X2030752X.

[5] Wolfram Heineken, Marc Richter, and Torsten Birth-Reichert. “Energy-Efficient Train Driving Based on Optimal Control Theory”. In: *Energies* 16.18 (2023). ISSN: 1996-1073. DOI: 10.3390/en16186712. URL: https://www.mdpi.com/1996-1073/16/18/6712.

[6] E. Khmelnitsky. “On an optimal control problem of train operation”. In: *IEEE Transactions on Automatic Control* 45.7 (2000), pp. 1257–1266. DOI: 10.1109/9.867018.

[7] Michael Nold and Francesco Corman. “Increasing realism in modelling energy losses in railway vehicles and their impact to energy-efficient train control”. In: *Railway Engineering Science* 32.3 (Sept. 2024), pp. 257–285. ISSN: 2662-4753. DOI: 10.1007/s40534-023-00322-4. URL: https://doi.org/10.1007/s40534-023-00322-4.

[8] William J Davis et al. “The tractive resistance of electric locomotives and cars”. In: (1926).

[9] Alejandro Cunillera et al. “Assessment of the Worthwhileness of Efficient Driving in Railway Systems with High-Receptivity Power Supplies”. In: *Energies* 13.7 (2020). ISSN: 1996-1073. DOI: 10.3390/en13071836. URL: https://www.mdpi.com/1996-1073/13/7/1836.

[10] Phil Howlett. “An optimal strategy for the control of a train”. In: *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics* 31.4 (1990), pp. 454–471. DOI: 10.1017/S0334270000006780.

[11] Amie R. Albrecht et al. "Energy-efficient train control: From local convexity to global optimization and uniqueness". In: *Automatica* 49.10 (2013), pp. 3072–3078. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2013.07.008. URL: https://www.sciencedirect.com/science/article/pii/S0005109813003567.

[12] Jie Yang et al. "Energy-Efficient Speed Profile Approximation: An Optimal Switching Region-Based Approach with Adaptive Resolution". In: *Energies* 9.10 (2016). ISSN: 1996-1073. DOI: 10.3390/en9100762. URL: https://www.mdpi.com/1996-1073/9/10/762.

[13] Masafumi Miyatake and Hideyoshi Ko. "Optimization of Train Speed Profile for Minimum Energy Consumption". In: *IEEJ Transactions on Electrical and Electronic Engineering* 5.3 (2010), pp. 263–269. DOI: https://doi.org/10.1002/tee.20528. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/tee.20528. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/tee.20528.

[14] Shaofeng Lu et al. "Single-Train Trajectory Optimization". In: *IEEE Transactions on Intelligent Transportation Systems* 14.2 (2013), pp. 743–750. ISSN: 1558-0016. DOI: 10.1109/TITS.2012.2234118.

[15] E. Rodrigo et al. "Optimizing Electric Rail Energy Consumption Using the Lagrange Multiplier Technique". In: *Journal of Transportation Engineering* 139.3 (2013), pp. 321–329. DOI: 10.1061/(ASCE)TE.1943-5436.0000483. eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%29TE.1943-5436.0000483. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29TE.1943-5436.0000483.

[16] Yihui Wang et al. "Optimal trajectory planning for trains – A pseudospectral method and a mixed integer linear programming approach". In: *Transportation Research Part C: Emerging Technologies* 29 (2013), pp. 97–114. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2013.01.007. URL: https://www.sciencedirect.com/science/article/pii/S0968090X13000193.

[17] Yihui Wang et al. "Optimal trajectory planning for trains under fixed and moving signaling systems using mixed integer linear programming". In: *Control Engineering Practice* 22 (2014), pp. 44–56. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2013.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0967066113001792.

[18] K.K. Wong and T.K. Ho. "Coast control for mass rapid transit railways with searching methods". In: *IEE Proceedings - Electric Power Applications* 151 (3 2004), pp. 365–376. DOI: 10.1049/ip-epa:20040346. eprint: https://digital-library.theiet.org/doi/pdf/10.1049/ip-epa%3A20040346. URL: https://digital-library.theiet.org/doi/abs/10.1049/ip-epa%3A20040346.

[19] Ning Zhao et al. "An integrated metro operation optimization to minimize energy consumption". In: *Transportation Research Part C: Emerging Technologies* 75 (2017), pp. 168–182. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2016.12.013. URL: https://www.sciencedirect.com/science/article/pii/S0968090X16302674.

[20] Zhongbei Tian et al. “System energy optimisation strategies for metros with regeneration”. In: *Transportation Research Part C: Emerging Technologies* 75 (2017), pp. 120–135. ISSN: 0968-090X. DOI: `https://doi.org/10.1016/j.trc.2016.12.004`. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X16302522`.

[21] Carlos Sicre et al. “Modeling and optimizing energy-efficient manual driving on high-speed lines”. In: *IEEJ Transactions on Electrical and Electronic Engineering* 7.6 (2012), pp. 633–640. DOI: `https://doi.org/10.1002/tee.21782`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/tee.21782`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/tee.21782`.

[22] Xiang Li and Hong K. Lo. “An energy-efficient scheduling and speed control approach for metro rail operations”. In: *Transportation Research Part B: Methodological* 64 (2014), pp. 73–89. ISSN: 0191-2615. DOI: `https://doi.org/10.1016/j.trb.2014.03.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0191261514000484`.

[23] A.P. Cucala et al. “Fuzzy optimal schedule of high speed train operation to minimize energy consumption with uncertain delays and driver’s behavioral response”. In: *Engineering Applications of Artificial Intelligence* 25.8 (2012), pp. 1548–1557. ISSN: 0952-1976. DOI: `https://doi.org/10.1016/j.engappai.2012.02.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0952197612000437`.

[24] C. Sicre, A.P. Cucala, and A. Fernández-Cardador. “Real time regulation of efficient driving of high speed trains based on a genetic algorithm and a fuzzy model of manual driving”. In: *Engineering Applications of Artificial Intelligence* 29 (2014), pp. 79–92. ISSN: 0952-1976. DOI: `https://doi.org/10.1016/j.engappai.2013.07.015`. URL: `https://www.sciencedirect.com/science/article/pii/S0952197613001437`.

[25] Youneng Huang et al. “Optimization of Train Operation in Multiple Interstations with Multi-Population Genetic Algorithm”. In: *Energies* 8.12 (2015), pp. 14311–14329. ISSN: 1996-1073. DOI: `10.3390/en81212433`. URL: `https://www.mdpi.com/1996-1073/8/12/12433`.

[26] Kemal Keskin and Abdurrahman Karamancioglu. “Energy-Efficient Train Operation Using Nature-Inspired Algorithms”. In: *Journal of Advanced Transportation* 2017.1 (2017), p. 6173795. DOI: `https://doi.org/10.1155/2017/6173795`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1155/2017/6173795`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1155/2017/6173795`.

[27] Y.-G. Kim et al. “Operating speed pattern optimization of railway vehicles with differential evolution algorithm”. In: *International Journal of Automotive Technology* 14.6 (Dec. 2013), pp. 903–911. ISSN: 1976-3832. DOI: `10.1007/s12239-013-0099-7`. URL: `https://doi.org/10.1007/s12239-013-0099-7`.

[28] Richard Bellman. “On the Theory of Dynamic Programming”. In: *Proceedings of the National Academy of Sciences of the United States of America* 38.8 (1952), pp. 716–719. ISSN: 00278424, 10916490. URL: `http://www.jstor.org/stable/88493` (visited on 12/11/2025).

[29] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN: 9780486428093.

[30] Ronald A Howard. *Dynamic programming and markov processes.* 1960.

[31] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (Sept. 1983), pp. 834–846. ISSN: 2168-2909. DOI: 10.1109/TSMC.1983.6313077.

[32] Richard Sutton. "Learning to Predict by the Method of Temporal Differences". In: *Machine Learning* 3 (Aug. 1988), pp. 9–44. DOI: 10.1007/BF00115009.

[33] Christopher John Cornish Hellaby Watkins. "Learning from Delayed Rewards". PhD thesis. Cambridge, UK: King's College, May 1989. URL: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

[34] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach. Learn.* 8.3–4 (May 1992), pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696. URL: https://doi.org/10.1007/BF00992696.

[35] Nahid Parvez Farazi et al. *Deep Reinforcement Learning and Transportation Research: A Comprehensive Review*. 2020. arXiv: 2010.06187 [cs.LG]. URL: https://arxiv.org/abs/2010.06187.

[36] Mengying Shang, Yonghua Zhou, and Hamido Fujita. "Deep reinforcement learning with reference system to handle constraints for energy-efficient train control". In: *Information Sciences* 570 (2021), pp. 708–721. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.04.088. URL: https://www.sciencedirect.com/science/article/pii/S0020025521004291.

[37] Dahan Wang et al. "Energy-saving operation in urban rail transit: A deep reinforcement learning approach with speed optimization". In: *Travel Behaviour and Society* 36 (Apr. 2024), p. 100796. DOI: 10.1016/j.tbs.2024.100796.

[38] Philip G. Howlett and Peter J. Pudney. "Critical Speeds and Strategies of Optimal Type". In: *Energy-Efficient Train Control*. London: Springer London, 1995, pp. 137–156. ISBN: 978-1-4471-3084-0. DOI: 10.1007/978-1-4471-3084-0_8. URL: https://doi.org/10.1007/978-1-4471-3084-0_8.

[39] Ligang Cheng et al. "A train trajectory optimization method based on the safety reinforcement learning with a relaxed dynamic reward". In: *Discover Applied Sciences* 6.9 (Aug. 2024), p. 469. ISSN: 3004-9261. DOI: 10.1007/s42452-024-06159-8. URL: https://doi.org/10.1007/s42452-024-06159-8.

[40] Nadav Merlis. *Reinforcement Learning with Lookahead Information*. 2024. arXiv: 2406.02258 [cs.LG]. URL: https://arxiv.org/abs/2406.02258.

[41] Yunfeng Zhang et al. "Multi-step look ahead deep reinforcement learning approach for automatic train regulation of urban rail transit lines with energy-saving". In: *Engineering Applications of Artificial Intelligence* 145 (2025), p. 110181. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2025.110181. URL: https://www.sciencedirect.com/science/article/pii/S0952197625001812.

[42] Google. *Google Maps Platform Elevation API*. Accessed: 23 September 2025. URL: https://developers.google.com/maps/documentation/elevation.

[43] Roger W Sinnott. "Virtues of the haversine". In: *Sky and Telescope* 68.2 (1984), pp. 158–159.

[44] E. A. Nadaraya. "On Estimating Regression". In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142. DOI: 10.1137/1109020. eprint: https://doi.org/10.1137/1109020. URL: https://doi.org/10.1137/1109020.

[45] Geoffrey S Watson. "Smooth regression analysis". In: *Sankhyā: The Indian Journal of Statistics, Series A* (1964), pp. 359–372.

[46] E. García-Portugués. *Notes for Predictive Modeling*. Version 5.12.2. ISBN 978-84-09-29679-8. 2025. URL: https://bookdown.org/egarpor/PM-UC3M/.

[47] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[48] Evgeny Kuprikov et al. "Deep reinforcement learning for self-tuning laser source of dissipative solitons". In: *Scientific Reports* 12 (May 2022). DOI: 10.1038/s41598-022-11274-w.

[49] python-visualization. *Folium*. Version 0.11.0. Dec. 28, 2020. URL: https://python-visualization.github.io/folium/.

[50] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.