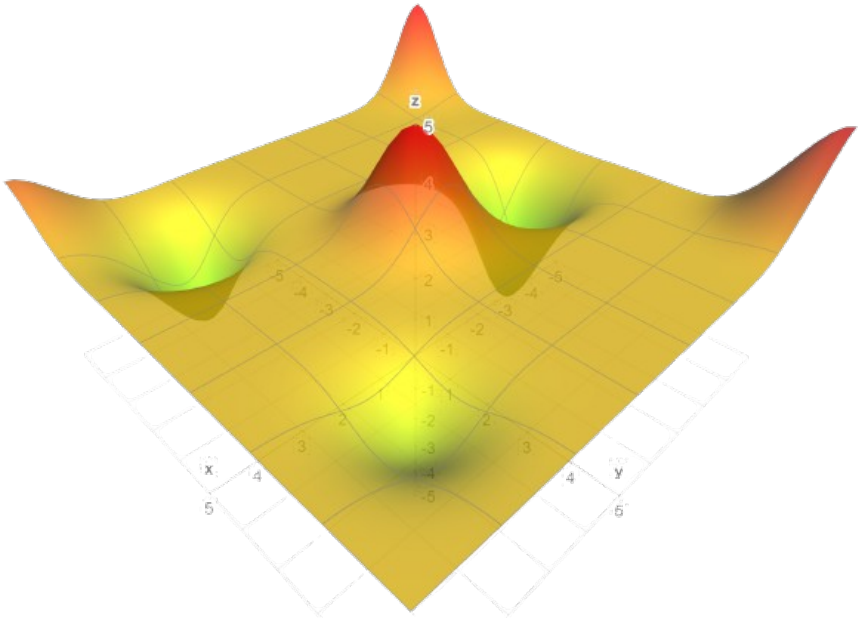
CHALMERS
UNIVERSITY OF TECHNOLOGY



Machine Learning Assisted Thermal Cost Functions for BEVs

An Inverse Reinforcement Learning (IRL) approach for cost function development

Master's thesis in Systems, Control and Mechatronics Programme

Prajwal Prashant Shetye

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Machine Learning Assisted Thermal Cost Functions for BEVs

An Inverse Reinforcement Learning (IRL) approach for cost function development

Prajwal Prashant Shetye

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Automatic Control Research Group
Chalmers University of Technology
Gothenburg, Sweden 2024

Machine Learning Assisted Thermal Cost Functions for BEVs
An Inverse Reinforcement Learning (IRL) approach for cost function development
Prajwal Prashant Shetye

University Examiner & Supervisor: Balázs Adam Kulcsár, Chalmers
Industry Supervisor: Bhavani Burugu, Volvo Car Corporation

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Automatic Control Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Simple non-convex non-linear cost function $z = c(x, y)$ visualization constructed in math3d.org to illustrate complexity of the problem.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Machine Learning Assisted Thermal Cost Functions for BEVs
An Inverse Reinforcement Learning (IRL) approach for cost function development
Prajwal Prashant Shetye
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis introduces a novel methodology employing machine learning, specifically inverse reinforcement learning (IRL), to develop a cost function represented using neural network and parameterized by its weights and bias for thermal management in BEVs. The purpose of this approach is to capture the intricate relationships between various operational variables and encode the correct task which is a challenge in practice. First, the study outlines the need of cost functions, and necessity to parameterize it. It then details the application of IRL to model these cost functions, which allows for the extraction of optimal strategies directly from data, circumventing the limitations of traditional modeling techniques. Secondly, the experimental setup, data collection, and implementation process of IRL are thoroughly described, highlighting how the model learns to predict and minimize thermal costs under various scenarios. The thesis concludes with a detailed discussion of methods to improve the cost function suggesting directions for further research. By leveraging machine learning for thermal management, this work contributes to the ongoing efforts to improve BEV performance, offering a scalable and efficient solution to one of the critical challenges in the field of electric vehicle technology.

Keywords:
cost function, optimization, machine learning, neural network, inverse learning, inverse optimal problem, inverse reinforcement learning, thermal management

Acknowledgements

I express my deepest gratitude to my academic supervisor, Balázs Adam Kulcsár, for the patient guidance, encouragement, and expertise provided in this project. Without the continuous support and interest, this thesis wouldn't have been the same.

The thesis work was carried out in the Thermal Control Team within the Thermal Management Department Volvo Cars, Gothenburg from January 2024 to June 2024. I would like to take this opportunity to thank everyone at Volvo Cars who has helped me in every step of the project.

I am thankful to my Manager Monica Johansson, for allowing me to work on this project. I am also grateful to my industry supervisor Bhavani Burugu, for providing insights and the freedom to carry out the work independently.

I am immensely grateful to, Yashasvi Nandivada, Babak Heydarnezhad, and Mukund Rudravajhala, for their advice that helped me to refine my direction through the course of this research. Our discussions around the subject have been thoroughly enlightening and encouraging.

Finally, I express my deepest gratitude to my mother Pranita Shetye for her unconditional love and trust in me, and my father Prashant Shetye for his support throughout my entire life; this dissertation stands as a testament to their unwavering faith in me.

Prajwal Prashant Shetye, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BEV	Battery Energy Vehicle
DQN	Deep Q-Network
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
EU	European Union
EV	Electric Vehicle
GCL	Guided Cost Learning
GPS	Guided Policy Search
IOC	Inverse Optimal Control
IRL	Inverse Reinforcement Learning
LP	Linear Programming
RL	Reinforcement Learning
MBMF	Model-Based DRL with Model-Free Fine-Tuning
MCTS	Monte Carlo Tree Search
ML	Machine Learning
MIMO	Multiple Input, Multiple Output
MPC	Model Predictive Control
PCA	Principal Component Analysis
PG	Policy Gradient
PID	Proportional, Integral, Derivative
PINN	Physics Inspired Neural Network
PPO	Proximal Policy Optimization
SARSA	State-Action-Reward-State-Action
SVM	Support Vector Machines
SysID	System Identification
TCP	Transmission Control Protocol
t-SNE	t-distributed stochastic neighbor embedding
VCC	Volvo Car Corporation
WLTC	Worldwide Harmonized Light Vehicles Test Cycle

Nomenclature

Below is the nomenclature mathematical notations that have been used throughout this thesis.

$\mathbf{x}$	Input vector
$\mathbf{y}$	Output vector
π	policy
t	Index for time step
$\mathbf{a}_t$	action vector at time t
$\mathbf{s}_t$	state vector at time t
$\mathbf{a}'$	probable future action at next step
$\mathbf{s}'$	future state as a consequence of $\mathbf{a}'$
$\mathbb{E}(x)$	Expected value of variable x
θ	neural network parameter vector of weights and biases
$\mathcal{L}_{\text{IOC}}(\theta)$	Loss term in IOC parameterized by θ
τ	trajectory
$p(\tau)$	probability of choosing τ
$c_\theta(\tau)$	cost associated with τ, parameterized by θ
Z	partition function Z, normalization factor
$\mathcal{D}_{demo}$	set of N trajectories obtained from demo
$q(\tau)$	Gaussian policy (background distribution) from which τ is sampled
$\mathcal{D}_{samp}$	set of M trajectories obtained from background distribution
$\hat{\mathcal{D}}_{\text{demo}}$	set of trajectories randomly sampled from $\mathcal{D}_{samp}$, $\hat{\mathcal{D}}_{\text{demo}} \subset \mathcal{D}_{\text{demo}}$
$\hat{\mathcal{D}}_{\text{samp}}$	set of trajectories randomly sampled from $\mathcal{D}_{samp}$, $\hat{\mathcal{D}}_{\text{samp}} \subset \mathcal{D}_{\text{samp}}$

Contents

List of Figures

List of Tables

1
Introduction

This chapter presents the section levels that can be used in the template.

1.1 Background and Motivation

In the EU transportation sector, approximately 71.7% of the total CO_2 emissions in 2019 are attributed to passenger vehicles [4]. Addressing this issue, the adoption of BEV stands out as a promising solution to mitigate these emissions [9]. Presently, lithium-ion (Li-ion) batteries dominate the EV industry due to their lighter weight, enhanced efficiency, higher energy density, and superior safety features compared to alternative battery technologies [5].
However, Li-ion batteries pose challenges such as high costs and sensitivity to heat. These issues not only lead to safety concerns, including the risk of thermal runaway but also impact energy density if the prescribed temperature range is not maintained. Consequently, battery lifespan and range of the vehicle are directly influenced [5]. Given the awareness of environmental safety and government regulations, competition among EV manufacturers is intense, making efficient thermal management of BEVs crucial from both safety and business perspectives. The choice and implementation of a control strategy are pivotal in this regard.
Despite the use of traditional control methods like bang-bang (on-off) and PID control in many vehicles today, predictive control is emerging with advantages such as MIMO capabilities and the ability to set constraints on variables [8]. Linear MPC is gaining popularity due to its relative simplicity. However, in practical applications, plant non-linearity poses challenges, requiring linearization of the nonlinear plant model at various operating points.
Thus this work proposes a novel approach using data-driven techniques for thermal management to circumvent the need to model the plant using first principles. There are several machine learning techniques most of which can be classified into supervised learning, unsupervised learning, and reinforcement learning. This work will explore reinforcement learning domain, in particular, as it shows great promise in this area due to its ability to learn optimal actions based on trial and error, effectively learning to balance between immediate cooling needs and long-term efficiency.

1.2 Purpose of the study

1.2.1 Scope

The purpose of the thesis work is to build and develop a framework for mode selection in thermal control systems that uses inverse reinforcement learning (IRL) to generate a parameterized cost function for optimal energy consumption. In particular, attention is paid to:

- Determine and select important factors and variables that affect thermal dynamics, eliminating irrelevant ones.
- Develop a machine learning algorithm to find the cost function for optimal thermal management.

1.2.2 Limitations

This work is subject to the following limitations:

- Only variables from the coolant circuit are used for the analysis, excluding climate and cabin components (refrigerant circuit).
- No representation of the state space of the plant model is performed either, excluding physically informed models.
- Policy optimisation and benchmarking are not included in the scope due to time constraints, but are explained in detail in further sections.
- Due to the non-linear and non-convex nature of the problem, it is not guaranteed that the global minima of the cost function are obtained.
- As the work is in initial stage, deployment on hardware is not guaranteed.

1.3 Ethical considerations

The integration of ML techniques for thermal management of BEVs raises a number of ethical considerations that must be addressed in order to align the research with broader societal values and principles. The utilisation of data-driven models in BEVs necessitates the prudent management of large data. In particular, issues of privacy are of paramount importance, especially when data may contain user-specific or geolocation information. It is of the utmost importance to implement robust techniques for anonymising data and to obtain the necessary consent from users in order to maintain privacy standards. In addition, data confidentiality agreements with the VCC ensure that all proprietary information and customer data used in this research are handled in strict compliance with legal and ethical standards. In addition, the environmental impact of these technologies must be carefully considered. While BEVs themselves reduce greenhouse gas emissions, it is important to optimise the entire lifecycle of these technologies, including energy consumed during data processing, model training and system operation, to minimise their overall environmental impact. Moreover, the use of sophisticated AI tools for language improvisation in this research is done with a commitment to responsible use. We make every effort to ensure that AI-enhanced content is accurate and adheres to ethical guidelines that prevent the dissemination of misleading or biased information.

2
Theory

2.1 Thermal Management System in BEV

The research was conducted using the VCC thermal management system, modeled in Simulink. The model integrates several components, including a battery, electric drives, a core computer, and actuators such as pumps, valves, and a radiator fan. In the scope of this project, the intricate thermal models of these individual components are not the primary focus, due to the adoption of a reinforcement learning approach. The reinforcement learning (RL) and inverse reinforcement learning (IRL) algorithms were implemented in Python. Consequently, to facilitate interaction with the Simulink model, an interface between the Python algorithms and the Simulink model was essential. Initially, this interface was implemented using the Matlab engine for Python, but due to the need for enhanced efficiency, a custom interface utilizing TCP sockets was later developed.
Furthermore, a custom environment compatible with OpenAI Gym was developed in Python, serving as a wrapper for this interface and, indirectly, the underlying Simulink model. This setup is referred to as the "plant" within this study, and the same is illustrated in Figure (2.1). Additional details on OpenAI Gym are available on their official website[18].
This configuration ensures that the developed framework remains compatible with nearly any other gym environment, should integration be necessary.

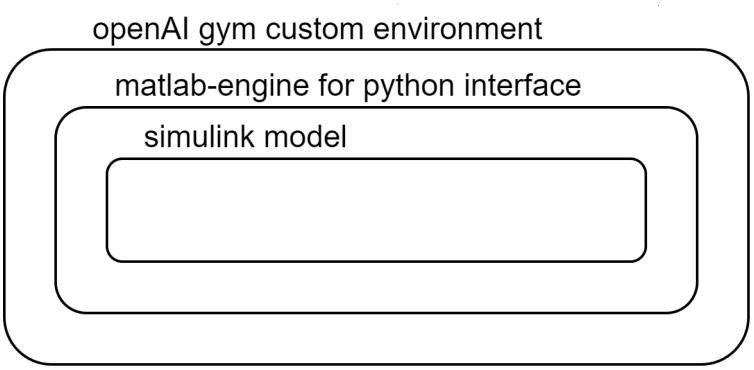


Figure 2.1: Illustration of abstractions within the plant.

2.2 Cost Function

The cost function serves as an essential mathematical tool for evaluating the "cost" or "penalty" associated with specific decisions, actions, or events within a system. It quantitatively assesses outcomes based on defined criteria, thereby facilitating a straightforward method to gauge performance. Cost functions can combine one or more objectives into a single function. This capability is particularly useful in systems where multiple aspects, such as energy utilization, performance, stability, and safety, need to be balanced. By integrating these variables into one function, it simplifies the optimization process across diverse goals. They also play a crucial role in informing decision-making processes. By evaluating the potential costs associated with different control actions, a cost function helps determine the most advantageous action based on the current state of the system. This is invaluable in control systems where optimal responses are essential for efficiency and effectiveness. Cost functions provide a common metric that can be used for benchmarking system performance. This evaluation tool is indispensable for comparing various strategies or configurations under uniform criteria, facilitating clear assessments of which methods meet or exceed performance standards.
Cost functions are represented by mathematical equations or through the use of universal function approximators like neural networks. **algebraic representation** employs explicit mathematical equations to define cost functions and is suitable for systems with well-understood variable relationships that can be accurately modeled mathematically. The primary advantage is its simplicity and interpretability. For systems with complex dynamics or poorly understood interactions among variables, a **representation using neural network** is employed. These networks can adeptly learn to approximate cost functions from data, effectively capturing non-linearities and intricate variable interactions that might elude traditional modeling approaches.
Cost functions can range from straightforward **convex** forms to highly complex **non-convex** structures, with significant challenges in optimization, especially in identifying the global minimum. A common analogy to understand this challenge is the "rolling ball analogy": imagine a ball rolling on the surface defined by the cost function. In a convex function, the ball will always settle at the lowest point, the global minimum. In contrast, with a non-convex function, the ball might easily get trapped in a lower area that is not the deepest point, representing a local minimum. Formally, non-convex functions often feature multiple local minima and saddle points, complicating the optimization efforts. Figure (2.2), highlights these complexities for both convex and non-convex cost functions with two input variables.

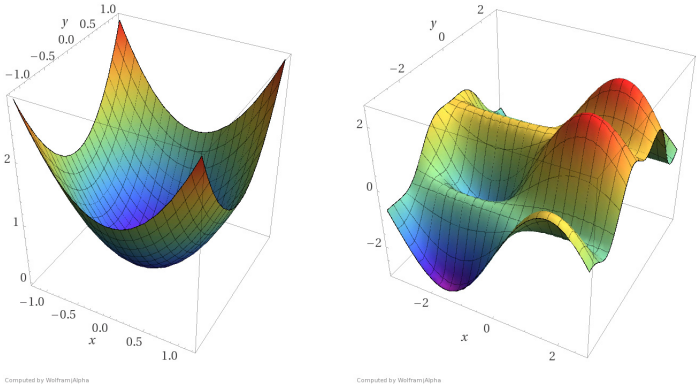


Figure 2.2: A visualization of convex (left) and non-convex (right) cost function with 2 input variables.[12]

2.2.1 Rationale for representation selected in this work

For this thesis, a neural network approach was chosen to represent the cost function due to the intricate behaviors associated with thermal management in BEVs. These behaviors involve multiple interacting variables that are not readily modeled using simple equations. By employing a neural network, these complex relationships can be learned directly from operational data, thereby enhancing the capacity to model and optimize the system's energy utilization efficiently, tailored to observed dynamics.
The cost function developed herein encompasses nine state variables and six action variables, cumulating in fifteen variables. This high dimensionality precludes straightforward visual illustration unlike the example illustrated in Figure (2.2). The high dimensionality also ensures that identifying the global minimum remains uncertain with the current state-of-the-art optimization techniques. This complexity reflects the broader challenges faced in optimization research, where high-dimensional spaces frequently test the limits of existing algorithms.

2.3 Machine Learning Approaches

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions.[19] In simple terms, given a set of input $\mathbf{x}$ and output $\mathbf{y}$, ML aims to obtain the function $\mathbf{y} = f(\mathbf{x})$ that generalizes the mapping for the given data. The difference between optimization and machine learning arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.[19] In general, ML techniques can be classified as: supervised, unsupervised or semi-supervised learning.[19] **Supervised Learning** majorly focuses on building a mathematical model of a set of labelled data that contains both the inputs and the desired outputs. The data is known as training data and consists of set of training examples. Some types of

supervised-learning algorithms include regression, classification, support vector machines (SVM) and decision trees. **Unsupervised Learning** studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data. Common unsupervised learning algorithms include clustering techniques like k-means, hierarchical clustering, and Gaussian mixture models, as well as dimensionality reduction techniques such as principal component analysis (PCA) and t-SNE. **Semi-Supervised Learning** falls between supervised and unsupervised learning and is used when the available data includes a large amount of unlabeled data and a small amount of labeled data. The goal in semi-supervised learning is to leverage the large pool of unlabeled data, along with the labeled data, to better learn the underlying structure in the data and improve the learning accuracy with less human effort in labeling data than would be required in a fully supervised setting.

2.4 Reinforcement Learning

It is a type of learning where an agent learns to make decisions by interacting with an environment. In RL, there is no explicit "correct answer" but it aims to find a policy π to control an agent in a environment [1], based on the feedback obtained in the form of rewards from the reward or cost function. A generic structure for reinforcement learning is illustrated in Figure (2.3). As per [3], RL is different from supervised learning. In interactive problems it is often impractical to obtain examples of desired behaviour that are both correct and representative of all the situations in which the agent has to act. In uncharted territory—where one would expect learning to be most beneficial—an agent must be able to learn from its own experience. Although one might be tempted to think of RL as a kind of unsupervised learning because it does not rely on examples of correct behaviour, RL is trying to maximize a reward signal unlike unsupervised learning which is trying to find hidden structure. Uncovering structure in an agent's experience can certainly be useful in RL, but by itself does not address the RL problem of maximizing a reward signal. [3] therefore considers RL to be third ML paradigm. This can also be justified by highlighting one of the challenges that arise in RL, and not in other kinds of learning, is the trade-off between exploration and exploitation.

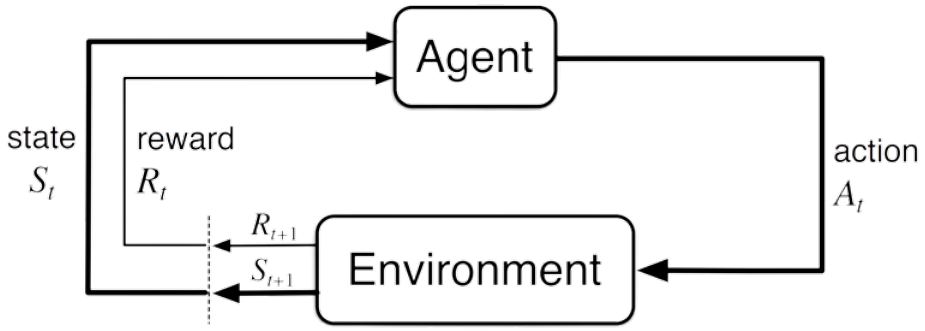


Figure 2.3: The agent–environment interaction in reinforcement learning.[3]

A few basic terminologies that are important in the context of this thesis are explained below[3]. A **state** is a formal description that provides complete information about the present status of the process or system sufficient to take control decisions. It is denoted by $\mathbf{s}_t$. The state is what the decision is made upon in each step of the learning process.
An **action** is an operation or decision that affects the state of the environment. It is denoted by $\mathbf{a}_t$. Actions are the means through which the agent interacts with the environment, with the goal of altering states to maximize cumulative rewards or minimize costs.
An **environment** in RL refers to the external process or system or the world that dynamically responds to the input stimulus or action. The response is a state at time $\mathbf{s}_{t+1}$ or observation at time $\mathbf{s}_{t+1}$ used to infer the state at time $\mathbf{s}_{t+1}$ and a reward or cost at time t denoting the quality of the taken action at time t given the state at time t. The environment defines the dynamics of how state transitions occur from $\mathbf{s}_t \rightarrow \mathbf{s}_{t+1}$ and how rewards are assigned.
A **model** in reinforcement learning context is an optional component that mimics the behavior of the environment. It predicts how the environment will respond to an agent's actions, providing predicted next states and expected rewards for state-action pairs. Models are used in model-based reinforcement learning methods to simulate and plan ahead, as opposed to model-free methods where the agent learns solely from interactions with the environment (discussed later). The model may be deterministic or stochastic. In a deterministic model the next state is solely a function of action taken, where as in a stochastic model the next state depends on some predefined probability distribution. Models are used for planning, that implies any way of deciding on a course of action by considering possible future situations before they are actually experienced.
An **agent** in RL is the entity that learns or makes decision and posses capabilities to interact with the environment. The agent makes decisions by choosing actions based on its policy (strategy), with the goal of maximizing cumulative rewards (or minimizing the cumulative costs) from the environment. The agent updates its policy based on the feedback (rewards or costs) received and the new states it encounters.[2]
A **policy function** or simply a policy denoted as π, is a strategy used by the agent to decide the next action based on the current state. It maps states of the

environment to actions the agent should take when in those states. Policies can be deterministic, where they specify a single action for each state, or stochastic, where they provide probabilities for selecting each possible action.
A **reward or cost function** in reinforcement learning assigns a scalar feedback signal to each state-action pair, which the agent uses to learn policies that maximize cumulative future rewards. In contrast, a cost function signals a penalty or cost associated with certain state-action pairs, guiding the agent to minimize the incurred costs. The reward or cost function is fundamental to defining the goals or objective of a reinforcement learning problem.

2.5 Types of Reinforcement Learning Algorithms

Most reinforcement learning (RL) techniques are generally classified as either model-based or model-free approaches[3]. A visual representation of this taxonomy is illustrated in Figure (2.4).

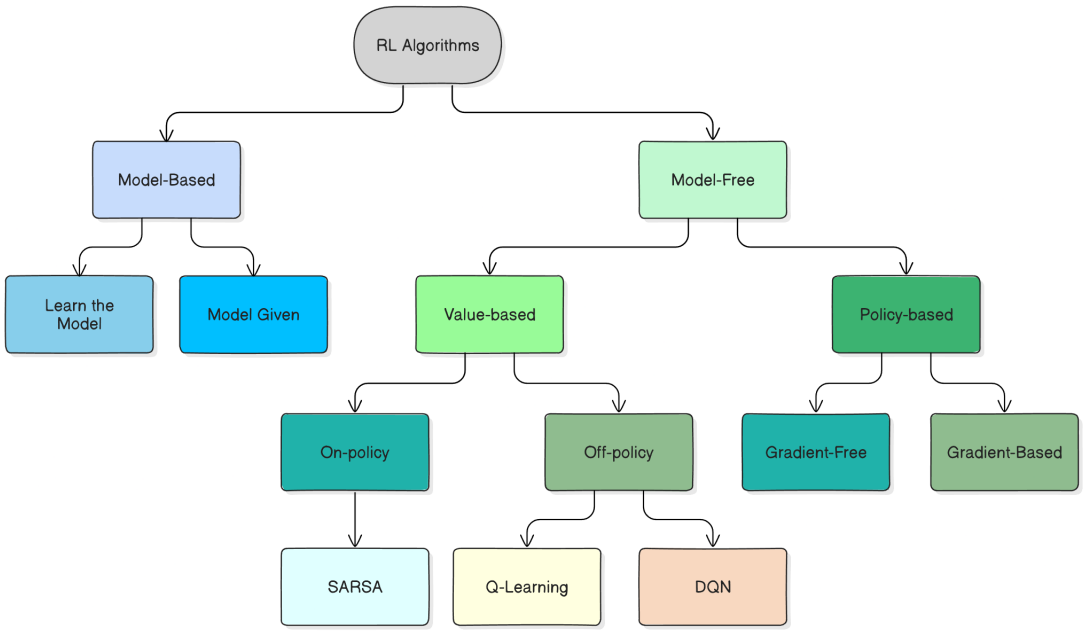


Figure 2.4: A non-exhaustive taxonomy of RL algorithms.[6]

2.5.1 Model-Free Reinforcement Learning

Model-free RL methods operate without an explicit model of the environment. These algorithms learn directly from experiences gathered through interaction with the environment, relying on trial and error to make decisions. The primary advantage of model-free methods is their simplicity and straightforward implementation, as they do not require the construction and maintenance of a model. Model-free methods can be further classified into Value-based and Policy-based methods.
Value-based methods focus on learning the value of actions to select the best one. They are further categorized as On-policy and Off-policy methods. **On-policy**

methods learn a policy by using data collected from running the same policy that is being learned. This means the learning algorithm evaluates and improves the policy based on the actions taken by the policy itself. The key characteristic of on-policy learning is that the decisions made by the policy influence the future data it receives for learning, thereby directly affecting the policy's development. Example: SARSA (State-Action-Reward-State-Action) is a classic example of an on-policy method. In SARSA, the agent learns a policy that decides the next action based on the current policy and updates its value estimates based on the transitions (state, action, reward, next state, next action) it experiences according to this same policy. **Off-policy methods**, on the other hand, learn a policy using data collected from running a different policy, known as the behavior policy, while the policy being learned is called the target policy. This separation allows the learning process to use data gathered from past policies or even from exploratory or random policies. Off-policy methods are powerful because they can learn from experiences that are off the current policy path, including old data or data collected from different strategies. Example: Q-Learning is one of the most well-known off-policy methods. In Q-Learning, the agent learns the optimal policy indirectly by learning the value of the optimal action to take in each state (regardless of the action taken by the policy that generated the data). This allows the policy to be updated from experiences that are not generated by the policy itself.
Policy-based methods directly learn the policy that maps states to actions. They are further categorized as Gradient-free and Gradient-based methods. **Gradient-free methods**, also known as derivative-free methods, optimize the policy without calculating gradients. These methods are particularly useful in situations where the objective function is not differentiable, is noisy, or the gradient is difficult to compute. Instead of using gradient descent, these methods typically explore the solution space by sampling and rely on other strategies like evolutionary algorithms, random search, or simple comparative evaluations. **Gradient-based methods** rely on the gradient (derivatives) of the performance measure with respect to the policy parameters to find the optimal policy. These methods are efficient in terms of convergence and are widely used when the gradient can be calculated directly or estimated reliably. They adjust the parameters in the direction that maximally improves the performance, as dictated by the gradient. Examples include: Policy Gradient, Proximal Policy Optimization (PPO).

2.5.2 Model-Based Reinforcement Learning

In contrast, model-based RL algorithms utilize a model of the environment, which they either learn or are provided a priori. These models predict the next state and the expected reward for state-action pairs, enabling the agent to plan by simulating future states and making informed decisions based on these predictions. Model-based methods can be further classified depending on whether the model is given or is learnt. In methods where the focus is to - **learn the model**, the agent learns a model of the environment and uses it to plan. The MBMF work is one such example that explores utilizing Model Predictive Control (MPC) with learned environment models[16]. In methods where the **model is given**, it is used directly for planing

the next course of actions. Examples include: Dyna-Q, Monte Carlo Tree Search (MCTS). Model-based approaches allow the agent to simulate outcomes of different actions and devise strategies that optimize expected rewards, often leading to more efficient learning processes compared to model-free methods.
In this study, model-free reinforcement learning methods, such as Deep Q-Networks (DQN), and gradient-based approaches, specifically Guided Policy Search (GPS), are employed and recommended for various tasks. DQN is utilized to generate the necessary trajectory data. Conversely, GPS is proposed as an effective technique for policy optimization. Detailed discussions on both of these methodologies are provided in subsequent sections.

2.6 Deep Q-Learning or Deep Q-Network (DQN)

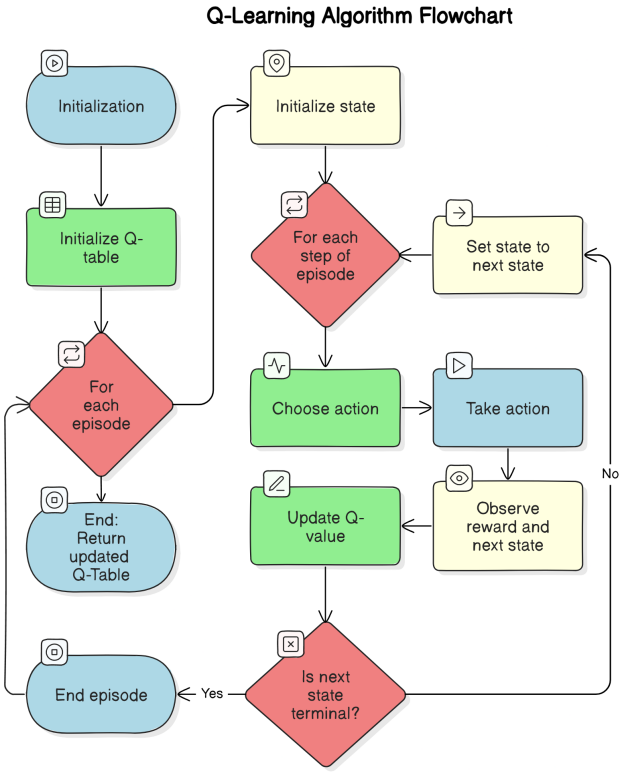


Figure 2.5: The Q-learning Algorithm.

2.6.1 Introduction to Q-Learning

Q-Learning is a model-free reinforcement learning algorithm that teaches an agent to act optimally by learning the value of actions directly from experience. Hence, it is categorized as a value-based, off-policy method since it focuses on finding the value of the action taken. It operates on the principle of learning an action-value function, known as the Q-function, which evaluates the worth of taking a specific

action in a given state. This function is updated using the formula:

$$\underbrace{Q(\mathbf{s},\mathbf{a})}_{\substack{\text{New}\\ \text{Q-value}}} \leftarrow \underbrace{Q(\mathbf{s},\mathbf{a})}_{\substack{\text{Old}\\ \text{Q-value}}} + \underbrace{\alpha}_{\substack{\text{Learning}\\ \text{Rate}\\ (0\sim1)}} \underbrace{\left[\underbrace{r}_{\text{Reward}} + \underbrace{\gamma}_{\substack{\text{Discount}\\ \text{Rate}\\ (0\sim1)}} \overbrace{\max_{a'} Q(s',a')}^{\substack{\text{max. of estimated Q-value}\\ \text{in the next transition}\\ \text{subject to next action}}} - \underbrace{Q(\mathbf{s},\mathbf{a})}_{\substack{\text{Old}\\ \text{Q-value}}} \right]}_{\text{TD error}}$$

where, r is the reward received after taking action $\mathbf{a}$ in state $\mathbf{s}$, $\mathbf{s}'$ is the subsequent state, $\mathbf{a}'$ is a potential future action, γ is the discount factor that determines the importance of future rewards compared to immediate rewards, and α is the learning rate, specifying the extent to which the newly acquired information will replace the old information. The entire algorithm for Q learning is summarized as a flow chart in Figure (2.5).

The Q-function is usually implemented as a 2D-Table popularly known as Q-Table with state s and action a as its row and column index of the cell in the table and each cell representing the Q-value. This is illustrated in Figure (2.6) Q-Learning is particularly effective in environments with discrete state and action spaces, where it balances exploration of new actions with exploitation of known actions to maximize rewards. This balance is often managed by strategies such as ϵ-greedy, enhancing the agent's ability to explore and exploit efficiently.

2.6.2 Extension to Deep Q-Learning (DQL)

While Q-Learning provides a robust framework for reinforcement learning, due to use of Q-Table, its application is limited when it comes to environments with large or continuous state spaces, such as those encountered in video games or robotic control. To address these challenges, Deep Q-Networks (DQN) extend Q-Learning by integrating deep neural networks to approximate the Q-function instead of using a Q-Table. An illustration of DQN and how it differs from Q-Table can be found in Figure (2.6).

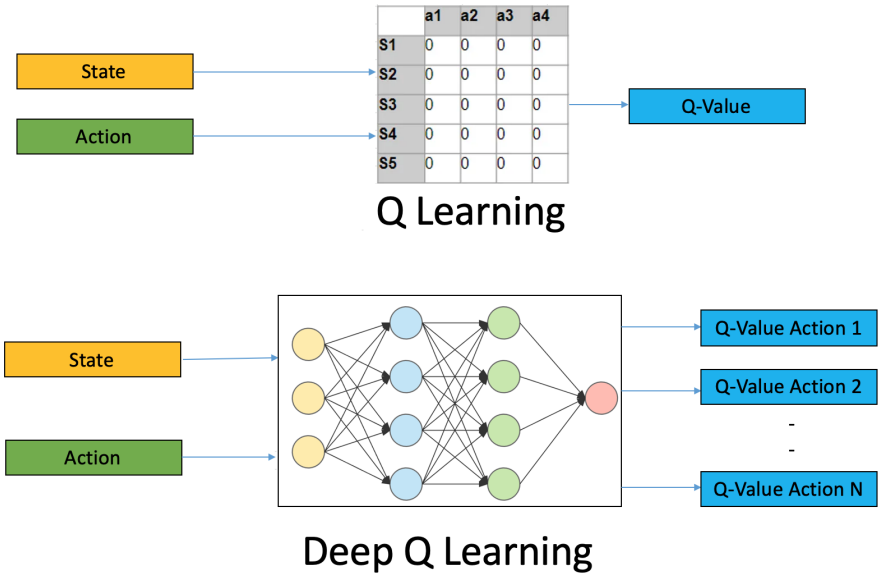


Figure 2.6: An illustration of Q-Table and Deep Q-Table.[10]

DQN enhances Q-Learning illustrated in Figure (2.5) by using a deep neural network that inputs the state of the environment and outputs a Q-value for each possible action. This approach allows handling of high-dimensional state spaces. Key innovations of DQN include[10] experience replay and target network. **Experience Replay** stores the agent's experiences at each time step, described by the tuple $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$, in a replay buffer. The network then samples mini-batches of these experiences to update the model, which helps in stabilizing the learning process by breaking the correlation between consecutive samples. **Target Network** represented using a DQN, utilizes a second neural network, which updates less frequently to provide stable learning targets. This dual-network architecture reduces the oscillations and divergences that are common in single-network setups during training.

2.6.3 Application in trajectory generation of this work

In this research, DQN is employed to autonomously generate demonstration, which is a collection of trajectories, sequences of states and actions pair that represent effective strategies or desired behaviour for navigating the environment. This capability is crucial in complex environments where manual demonstration is impractical. By comparing the trajectories generated by Q-Learning using DQN, one can evaluate the impact of integrating deep learning into classic reinforcement learning frameworks, particularly in terms of handling complex decision-making scenarios.
The integration of Q-Learning with deep neural networks in DQN represents a significant advancement in reinforcement learning, allowing for more effective handling of complex and high-dimensional environments. This development not only extends the applicability of Q-Learning but also enhances ability to generate sophisticated strategies that can be used in advanced simulations and real-world applications, thus broadening the scope and impact of this research.

2.7 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a specialized area of machine learning that focuses on deducing the reward function or cost function of an agent by observing its behavior over time. Unlike traditional reinforcement learning, which aims to optimize the policy based on a known reward function, IRL seeks to infer what objectives the agent is optimizing by analyzing its actions within a given environment. The assumption behind IRL is that the observed behaviors are generated by a rational agent striving to maximize a reward function. However, since the reward function is unknown, the task of IRL is to reverse engineer this function based on the observed state-action trajectories. Mathematically, it can be seen as solving the inverse problem of reinforcement learning, where the aim is to recover a reward function from observed policies rather than derive policies from a given cost function or reward function.

2.7.1 IRL Methods

Several approaches have been developed to solve the IRL problem, ranging from direct methods that assume specific forms for the reward function to more complex approaches that use deep learning for function approximation. Prominent methods include[7] Linear Programming, Bayesian Methods, Deep IRL.

Linear Programming (LP) in the context of IRL is used to solve the optimization problem where the reward function is assumed to be a linear combination of known features. It is particularly effective when the structure of the reward function is simple and when constraints and objectives can be linearly defined. It provides exact solutions and is computationally efficient for smaller, well-defined problems. But it is not suitable for complex reward structures or when the relationships between variables are non-linear.

A class of methods categorized as **Bayesian methods** approaches IRL using probabilistic methods to infer the reward function by treating it as a random variable with a prior distribution. These methods update the belief about the reward function based on the observed behavior using Bayesian inference. They incorporates uncertainty and prior knowledge effectively, offering a robust framework for dealing with incomplete or noisy data. It can be computationally intensive, especially with large state spaces or complex priors.

Deep IRL extends traditional IRL methods by leveraging neural networks to model the reward function, allowing for handling of high-dimensional and complex environments. They are capable of learning nuanced, non-linear reward functions that traditional linear models cannot. But these methods requires large amounts of data and significant computational resources; may be prone to over-fitting without proper regularization.

IRL has wide applications across various fields. In robotics, it is used for understanding and replicating human behavior in robots. In economics, Modeling decision-making processes of agents. In Autonomous Vehicles, decoding driving behaviors based on observed actions of drivers.

2.7.2 Comparison of IRL with Inverse Optimal Control

Both Inverse Reinforcement Learning (IRL) and Inverse Optimal Control (IOC) are concerned with determining the underlying objectives that explain observed behaviors or trajectories. In both frameworks, the primary goal is to infer a cost or reward function that an agent, whether a physical system or a software entity, seems to be optimizing. Both methods utilize observed data to reverse engineer this function, making them crucial in fields where understanding or replicating complex decision-making is necessary. While IRL and IOC share conceptual similarities, they diverge significantly in their application domains and underlying assumptions.
IOC traditionally finds its roots in control theory, where it is often applied to problems involving physical systems and their controls. It is particularly prevalent in engineering disciplines where the dynamics of the system are well-understood and modeled. In contrast, IRL is more commonly found in the context of artificial intelligence and machine learning, dealing with agents whose decision-making logic needs to be deciphered or replicated.
IOC often operates under the assumption that the dynamics of the system are known and focuses on deducing the cost function used within these dynamics. On the other hand, IRL can be applied even when the system dynamics are unknown or partially known, relying on statistical and machine learning techniques to uncover the reward structure guiding the agent's behavior.
In IOC, the problem is generally framed as a control problem where the optimal control actions are derived from the Bellman equations typical in dynamic programming. IRL, however, does not necessarily assume the existence of a control system and focuses more broadly on any decision-making process, utilizing algorithms from the broader field of machine learning to estimate the reward function.
IRL, especially when combined with modern deep learning approaches, offers greater flexibility in handling high-dimensional and complex environments compared to traditional IOC methods, which might require precise models of the environment to be effective.

2.7.3 IRL for obtaining thermal cost functions in BEVs

In this research, IRL is utilized to determine the thermal cost functions for optimal energy consumption in BEVs. By analyzing the trajectories of BEVs under various thermal conditions and operational scenarios, IRL helps in deducing the implicit cost functions that drivers or automated systems optimize for energy efficiency and safety. Specifically, Deep IRL is used where the neural networks model the cost function. Since there are many methods within the Deep IRL approach, this work will focus on GCL for cost function deduction and later for optimization.
The decision to use IRL stems from its ability to model complex decision-making processes without requiring a predefined reward structure. This is particularly advantageous in thermal management, where multiple interacting factors influence system performance and user comfort. Applying IRL allows for the extraction of nuanced insights into how energy consumption can be optimized through adaptive thermal management strategies, potentially leading to more efficient BEV designs and operation guidelines.

IRL offers profound insights into understanding the underlying motivations behind observed behaviors in agents. It bridges the gap between merely mimicking behavior and understanding the fundamental reasons for such behaviors, providing a deeper level of insight into decision-making processes.

2.8 Guided Cost Learning (GCL)

Guided Cost Learning (GCL) is an advanced algorithm within the field of Inverse Reinforcement Learning (IRL) that employs policy optimization techniques to effectively learn cost functions from demonstrations. GCL is particularly suitable for environments characterized by high-dimensional continuous systems. It is designed to address some common challenges in Inverse Optimal Control (IOC), such as the need for engineered features and the difficulty of dealing with complex dynamic systems. Most of the theoretical foundations in this section is heavily inspired from the previous research on Guided Cost Learning (GCL) as presented in [13]. GCL integrates elements of policy optimization with principles of maximum entropy IRL. This section introduces few essential concepts, attempts to present a simplified overview of the same and explains their relevance in learning from expert demonstrations.
The **maximum entropy** approach to IRL posits that among all possible explanations for observed behavior, the one that maximizes entropy—or randomness—is preferred. This method models the expert's actions as a probability distribution that not only maximizes rewards but also promotes diversity in actions. It selects the least biased model among all that fit the observed behavior, thereby addressing multiple challenges:

- **Generalization**: This principle supports generalization to unseen scenarios by avoiding strong assumptions about the reward structure, thereby enhancing the adaptability of learned models.
- **Ambiguity Resolution:** MaxEnt helps resolve the ambiguity in demonstration data by preferring the most uniform distribution over behaviors consistent with observed actions, effectively avoiding over-fitting.
- **Noise and Imperfection Tolerance**: By incorporating stochasticity, MaxEnt robustly handles noise and imperfections in demonstration data, reflecting realistic variations in agent behavior.
- **Practical Implementation:** MaxEnt provides a clear optimization framework, making it practically implementable, especially in environments with large or continuous state spaces.
- **Compatibility with Probabilistic Models:** It aligns well with probabilistic graphical models, allowing for the integration of additional constraints or prior knowledge.

[11] provides a concise summary of the mathematical foundations, while [15] offers an extensive exploration of the maximum entropy principle in greater detail.
The **partition function** Z is crucial in the context of maximum entropy methods as it ensures the probabilities of all possible trajectories sum to one. It is defined as:

$$Z = \int \exp(-c_\theta(\tau))d\tau, \tag{2.1}$$

This function integrates the exponential negative costs over all possible trajectories, but calculating it directly is often computationally infeasible in continuous or large state spaces.

Adaptive sampling in GCL refers to the technique of dynamically adjusting the sampling distribution of trajectories based on the evolving estimates of the cost function. This method helps focus computational resources on the most informative regions of the state-action space, thereby improving the efficiency and accuracy of learning.

2.8.1 Algorithmic Implementation

GCL iteratively refines the cost function and the sampling distribution using trajectory samples. This is achieved through the following algorithms, which detail the process of updating and optimizing these elements. The primary algorithm of GCL alternates between sampling the trajectories and optimizing the cost function and is illustrated in algorithm (1). Importance weights are used during cost optimization to correct sampling bias, ensuring that the trajectories used reflect the actual distribution under the current cost function. The algorithm for cost optimization is represented using algorithm (2).

Algorithm 1 Guided cost learning

Initialize $q_k(\tau)$ as either a random initial controller or from demonstrations
for iteration $i = 1$ to I **do**
    Generate samples $\mathcal{D}_{\text{traj}}$ from $q_k(\tau)$
    Append samples: $\mathcal{D}_{\text{samp}} \leftarrow \mathcal{D}_{\text{samp}} \cup \mathcal{D}_{\text{traj}}$
    Use $\mathcal{D}_{\text{samp}}$ to update cost c_θ using Algorithm (2)
    Update $q_k(\tau)$ using $\mathcal{D}_{\text{traj}}$ and the method from (Levine & Abbeel, 2014)[14] to obtain $q_{k+1}(\tau)$
end for
return optimized cost parameters θ and trajectory distribution $q(\tau)$

Algorithm 2 Nonlinear IOC with stochastic gradients to update the cost c_θ

for iteration $k = 1$ to K **do**
    Sample demonstration batch $\hat{\mathcal{D}}_{\text{demo}} \subset \mathcal{D}_{\text{demo}}$
    Sample background batch $\hat{\mathcal{D}}_{\text{samp}} \subset \mathcal{D}_{\text{samp}}$
    Append demonstration batch to background batch:
        $\mathcal{D}_{\text{samp}} \leftarrow \hat{\mathcal{D}}_{\text{demo}} \cup \hat{\mathcal{D}}_{\text{samp}}$
    Estimate $\frac{d\mathcal{L}_{\text{IOC}}}{d\theta}(\theta)$ using $\hat{\mathcal{D}}_{\text{demo}}$ and $\hat{\mathcal{D}}_{\text{samp}}$
    Update parameters θ using gradient $\frac{d\mathcal{L}_{\text{IOC}}}{d\theta}(\theta)$
end for
return optimized cost parameters θ

In Algorithm (2), the loss $\mathcal{L}_{\text{IOC}}(\theta)$ is a negative log-likelihood corresponding to the

IOC model is given by:

$$\mathcal{L}_{\text{IOC}}(\theta) = \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{\text{demo}}} c_\theta(\tau_i) + \log Z \tag{2.2}$$

where, Z is difficult to compute for large or continuous domains, and presents the main computational challenge in maximum entropy IOC and hence is approximated as shown below for M trajectories in $\mathcal{D}_{\text{samp}}$.

$$Z \approx \frac{1}{M} \sum_{\tau_j \in \mathcal{D}_{\text{samp}}} \frac{\exp(-c_\theta(\tau_j))}{q(\tau_j)} \tag{2.3}$$

2.8.2 Application in Thermal Cost Function Deduction

In this research, GCL is utilized to deduce thermal cost functions for optimal energy consumption in Battery Electric Vehicles (BEVs). The algorithm's capability to adaptively sample and refine cost functions based on observed data makes it ideal for optimizing complex systems like thermal management in BEVs.
Step 6 in Algorithm (1) is also known as the policy optimization step and is skipped in this work due availability of time. A dedicated section on the same is presented in later sections. Figure (3.2) illustrates the combined Algorithm (1) and (2) without the intricate mathematics.
Overall, GCL offers a robust framework for learning complex system behaviors. By combining policy optimization with maximum entropy principles, it efficiently handles high-dimensional environments and improves the understanding of expert behavior through learned cost functions. But this work only focuses on cost function deduction and not on policy optimization for reasons mentioned previously.

3
Methods

3.1 Developing the custom environment

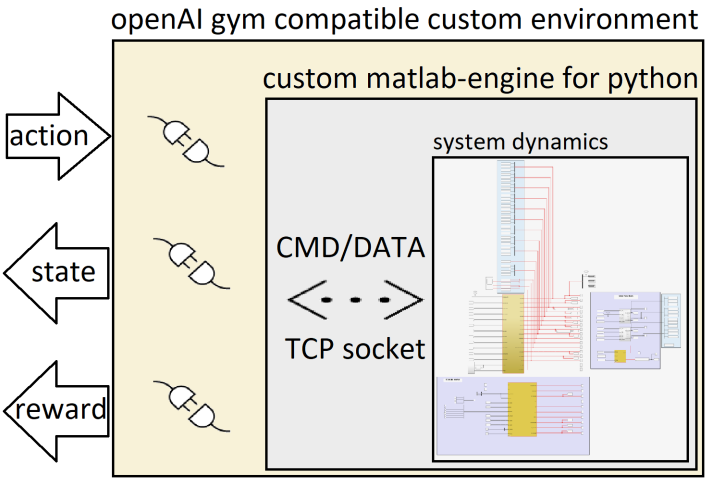


Figure 3.1: A detailed illustration of the custom openAI gym environment (plant)

3.1.1 Need for a custom environment

In traditional control system methodologies, a transfer function or a state-space representation is typically required to describe the plant. For systems characterized by a relatively low number of variables n in the state vector $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$, this modeling process can be straightforward. However, for larger, more complex systems, the development of such models often necessitates the use of system identification (SysID) techniques or universal function approximators such as neural networks.

Contrastingly, the approach in this thesis leverages a reinforcement learning (RL) framework, which inherently simplifies some aspects of this modeling process. Given the availability of a Simulink model of the thermal system—which should not be confused with the term 'model' as used in RL terminology, the traditional methods of SysID and neural network modeling are circumvented. In the context of RL, the primary requirements are the state of the system and the associated reward feedback, which are used by the RL agent to learn effective actions.

To integrate the Simulink model into the RL framework, an OpenAI Gym-compatible environment was custom-developed. This environment utilizes the Simulink model

to simulate the dynamics of the thermal system, providing the RL agent with necessary state and reward data. It is important to note that the Simulink model of the thermal system is proprietary to VCC, and as such, specific details regarding its configuration remain confidential and are outside the scope of this thesis. This exclusion does not affect the contribution of this work, as the focus remains primarily on the interaction between the RL agent and the environmental states and rewards, rather than on the underlying specifics of the thermal model itself.

3.1.2 Implementation of the Custom Environment

Table 3.1: Variables in the state vector

Variable in state vector	**# of discrete values for a given domain**
Coolant Temp at Inverter 1 inlet	501
Coolant Temp at ED 1 inlet	401
Coolant Temp at HV Battery inlet	201
Coolant Temp at Inverter 2 inlet	501
Coolant Temp at ED 2 inlet	651
Total power consumption	101
ED pump power consumption	46
Battery pump power consumption	46
Radiator fan power consumption	91

Table 3.2: Variables in the action vector

Variable in action vector	**# of discrete values for a given domain**
Valve A position	4
Valve B position	4
Valve C position	2
Battery circuit pump speed	10
ED circuit pump speed	10
Radiator fan speed	10

This sub-section details the creation of a custom environment which is illustrated in Figure (3.1), is designed to be compatible with the OpenAI Gym framework. This setup enables the application of various RL algorithms to control and optimize the thermal management in Battery Electric Vehicles (BEVs).
The core of the environment is the system dynamics, simulated using a Simulink model that represents the thermal behavior of the BEV. To facilitate run-time interaction between the RL algorithms and the Simulink model, a custom MATLAB-

engine for Python was developed using TCP sockets, replacing the standard MATLAB-engine due to its limitations in data transfer speeds. This custom engine significantly improves the efficiency of communication between the computational components. Communication between the Python environment and the Simulink model is managed via TCP sockets allowing execution of commands and exchange of data. This setup is called as custom implementation of matlab-engine for python in this work. It ensures robust and efficient data exchange, allowing the RL agent to continuously receive updated state and reward information necessary for run-time decision-making in dynamic control scenarios.
Integration with the OpenAI Gym framework is achieved by implementing essential methods that allow the RL agent to send actions to, and receive state and reward information from, the Simulink model. This interaction is critical for the RL agent's learning process, enabling it to make informed decisions that optimize the system's performance. Tables (3.1) and (3.2) defines the state vector and the action vector summarizing the variables in these vector. Overall, this custom environment expands the capabilities by enabling its use for RL applications.

3.2 Data Collection and Preparation

Training a cost function to capture the expected behavior, which in this instance is optimizing energy usage, necessitates recording data that exhibits the characteristics of this behavior.
As outlined in the Guided Cost Learning (GCL) Algorithm (1), demonstrations are crucial in the training process of the cost function. A demonstration, or demo, consists of a collection of trajectories. Each trajectory is a time series of state-action pairs $(\mathbf{s}_t, \mathbf{a}_t)$, representing intended, though not necessarily optimal, behavior. The primary purpose of a demo is to illustrate potential approaches to achieving the target goal, enabling the cost function to identify an optimal method. This process should not be confused with imitation learning, where the aim is to replicate the actions within the trajectory. In contrast, the goal in GCL is to capture the intentions behind the actions in the trajectories.
For data collection, a Deep Q-Learning (DQL) agent is trained across 75 episodes to serve as an initial, suboptimal controller. This agent generates actions based on the Q-learning algorithm, responding to the state $(\mathbf{s}_t)$ at each timestep (t). The state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$ is recorded at each timestep (t) as part of a trajectory, with each trajectory stored per episode. This process is repeated until 10 trajectories have been accumulated for the demonstration. The choice for selecting number of episodes and trajectories is completely random at this stage since the main focus of the work was to design and implement the complete workflow required to develop the cost function. The consequence of this is something which requires exploration.

3.3 Deducing the Cost Function

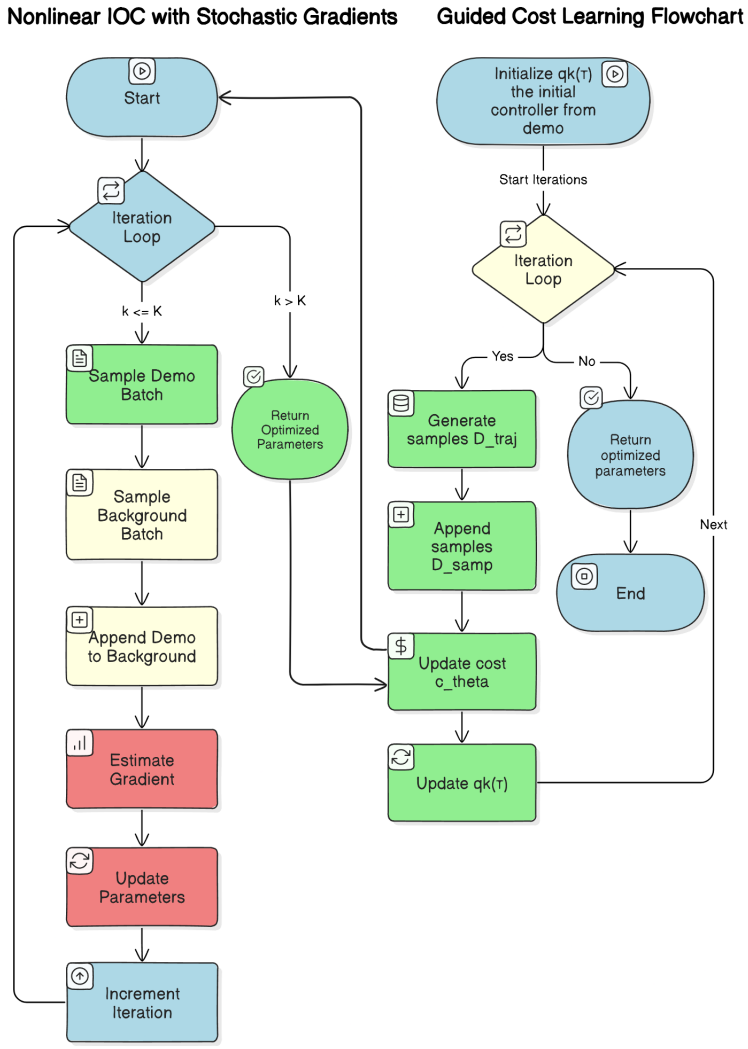


Figure 3.2: Complete and simplified flow chart of GCL algorithm

The architecture of the cost function in this research utilizes a neural network model. This network comprises an input layer with 15 neurons, corresponding to the number of variables in the combined state and action vectors. It features a single output neuron that represents the cost value, effectively outputting the computed cost for given inputs. The network includes one hidden layer containing 128 neurons, with all layers being fully interconnected. The ReLU activation function is employed for the hidden layers to introduce non-linearity, whereas the output layer does not use an activation function to directly represent the continuous cost value.
The neural network-based cost function is trained employing the Guided Cost Learning (GCL) Algorithm, referenced in (1) and (2). The combined algorithm without the sophisticated mathematics is illustrated as a flow chart in Figure (3.2). This training process makes use of the demonstrations outlined in the Data Collection and

Preparation section, ensuring that the model learns effectively from the trajectories that exhibit the desired behavior.
After several cycles or episodes of training, energy consumption is recorded to evaluate the performance of the trained model. The plots and analyses are detailed in the following subsections. It is important to note that these results pertain specifically to the simulation setup described herein. To generalize the applicability of the cost function across various scenarios, additional training on different simulation setups is recommended.

3.4 The Simulation Setup

This section describes the specific conditions under which the simulations were performed to evaluate the thermal management strategies in Battery Electric Vehicles (BEVs) using the developed cost function. Understanding these conditions is crucial as they significantly influence the behavior and outcomes of the simulations.

Ambient Temperature: The ambient temperature for all simulations is set to 26°C. This affects various thermal properties and behaviors of the vehicle's components and the decision making process for the choice of control actions.

Initial Conditions: At the start of each simulation, it is assumed that all components of the vehicle, are in equilibrium to the ambient temperature. This standardization of initial conditions ensures that the simulation results are not unduly influenced by variations in starting temperatures of different components.

Post Fast Charging Case: A critical initial condition set for the simulations involves the battery temperature. The initial temperature of the battery (T_{batt}) is set to 43°C. This condition is representative of a post fast-charging scenario where the battery tends to be significantly warmer than the ambient due to the heat generated from rapid charging processes. This scenario is particularly relevant as it poses a thermal management challenge, making it a crucial test case for evaluating the effectiveness of the proposed cost function.

Drive Cycle: The drive cycle used in the simulations is the Worldwide Harmonized Light Vehicles Test Cycle (WLTC3)[17]. This drive cycle is chosen because it realistically representative of vehicles driven in EU and Japan and includes a range of operating phases such as idling, acceleration, steady cruising, and deceleration. WLTC3 is structured to simulate urban, suburban, and highway driving, thereby providing comprehensive data on vehicle performance and energy utilization under varied driving conditions.

These simulation parameters are meticulously chosen to ensure that the results are robust, providing a reliable basis for evaluating the thermal management strategies developed in this thesis. Further analyses and results derived from these simulations are discussed in the subsequent sections, illustrating the impact and effectiveness of the optimized cost function under the defined setup.

4
Results and Discussions

4.1 Training Results

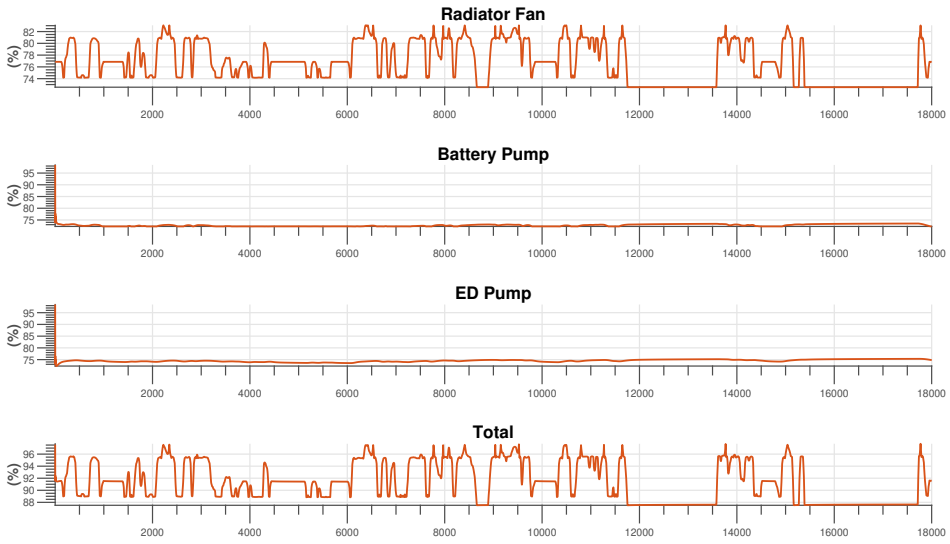


Figure 4.1: Energy consumption as a percentage of its maximum after training for episode 1

The training results are presented through Figures (4.2), (4.3), and (4.4), each illustrating the outcomes after a single WLTC cycle test run post-training for 1, 11, and 50 episodes respectively. The plots are divided into four subplots in a 4x1 format, detailing the energy consumption for the radiator fan, battery circuit pump, electric drive (ED) pump, and the total energy consumption of all tested components on y-axis and discrete time steps on x-axis where the step size is 0.1 second. It is important to note that the energy consumed by valve actuation is omitted from this analysis due to its negligible impact compared to the consumption by the pumps and the radiator fan.

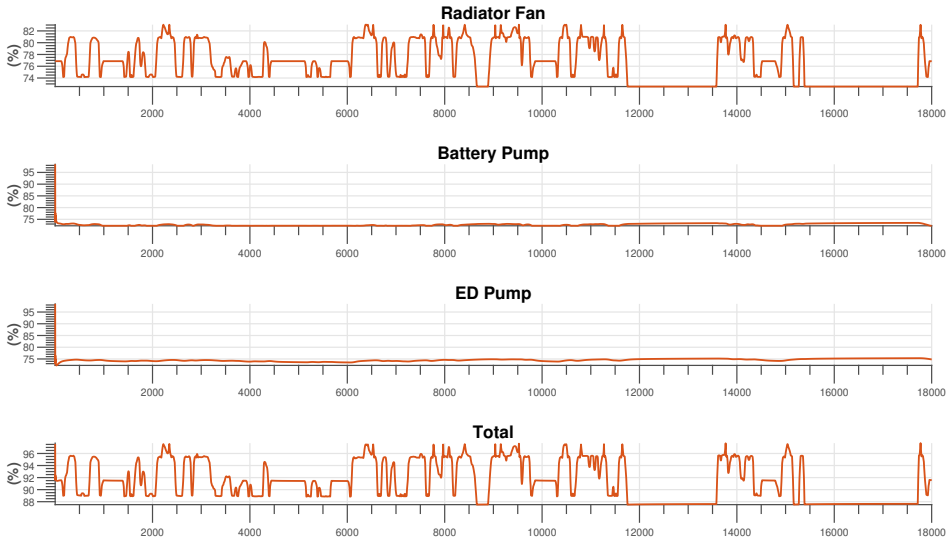


Figure 4.2: Energy consumption as a percentage of its maximum after training for episode 1

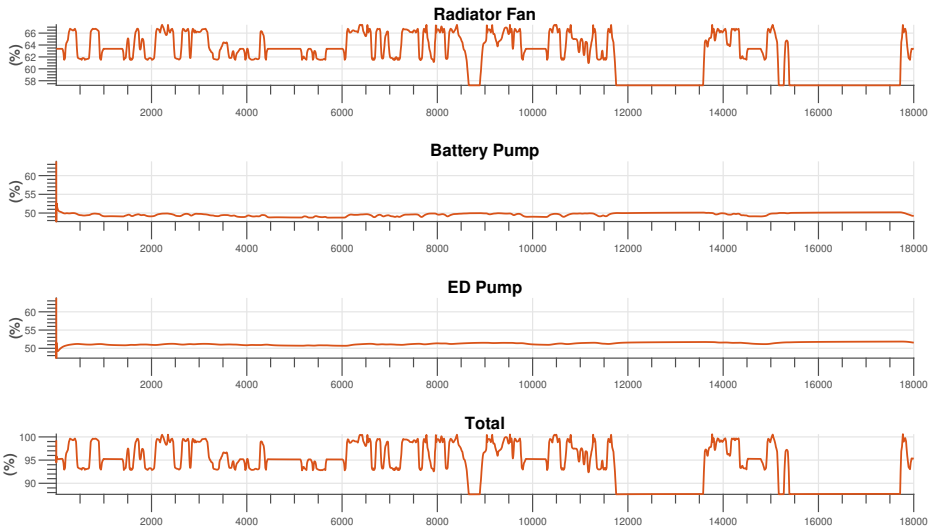


Figure 4.3: Energy consumption as a percentage of its maximum after training for episode 11

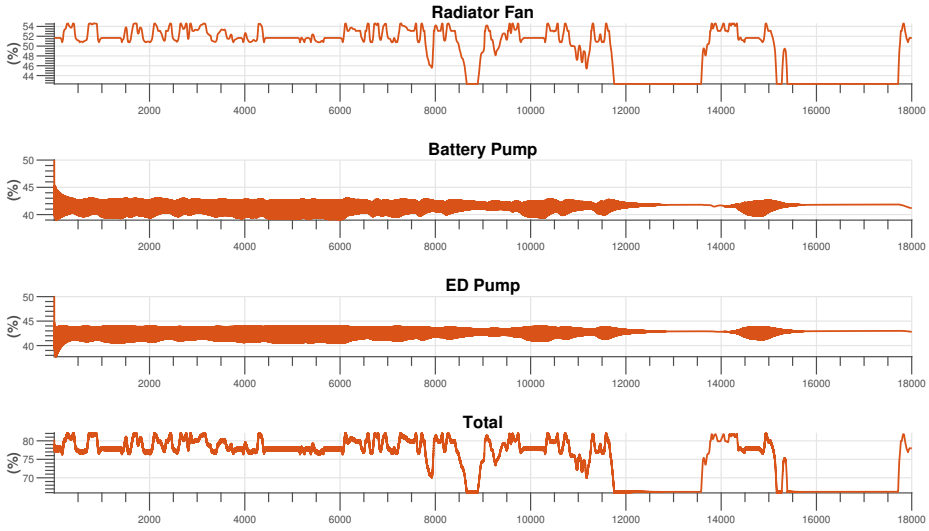


Figure 4.4: Energy consumption as a percentage of its maximum after training for episode 50

4.2 Observation and Discussion

This section explores the energy consumption trends and the enhancements in performance observed across various components of the vehicle's thermal management system. As depicted in the plots, the radiator fan exhibits higher and more variable energy usage compared to the pumps, underscoring its critical role in managing thermal loads. A notable reduction in the variability and peaks of energy consumption is observed with progression from 1 to 50 episodes, suggesting enhancements in the control strategies optimized for energy efficiency.
Significant reductions in operation costs were observed with an increase in training episodes, highlighting the neural network-based cost function's capability to enhance operational efficiencies. However, increased exploratory actions during training occasionally led to system crashes when such actions induced sudden and undesirable changes in the system. This necessitates a careful balance of exploration within the training regime to avoid instability.
During the 50th training run, significant fluctuations in energy consumption were noted. This could be attributed to an action-based policy that, within the context of the Guided Cost Learning (GCL) process, struggles with policy optimization due to the need for a Gaussian policy. This often results in erratic behavior under certain operational conditions.

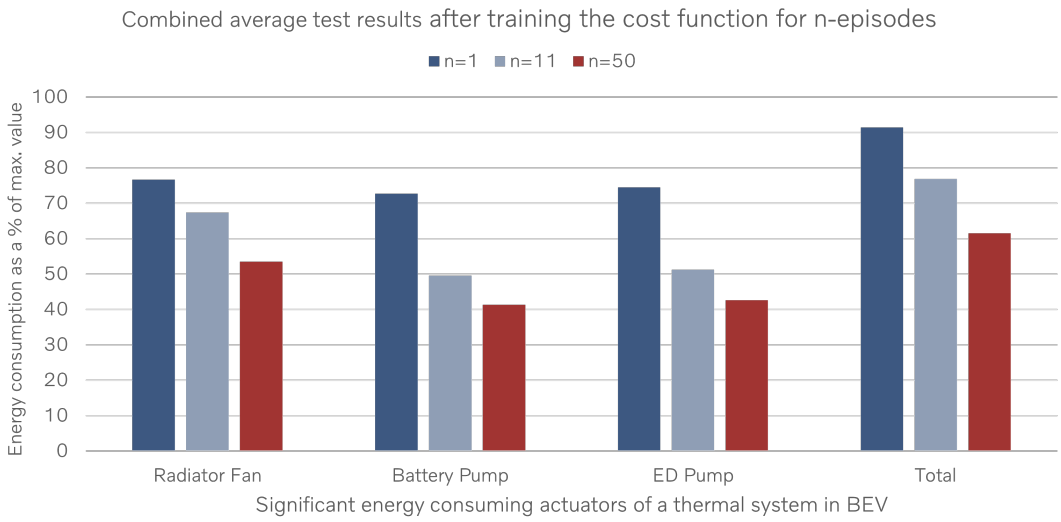


Figure 4.5: Combined average test results after training the cost function for n-episodes

Figure (4.5) summarizes the energy consumption trends following test runs after training the cost function for 1, 11, and 50 episodes. Each bar represents the average energy consumption of significant actuators within the BEV's thermal system, quantified as a percentage of their maximum potential value. The radiator fan, which has a considerable influence on total energy use, shows a consistent decrease in consumption over time. This pattern suggests that while initial reductions in energy consumption for the pumps are significant, the overall energy savings are influenced heavily by the radiator fan.
The analysis suggests that prolonged training could potentially result in further reductions in energy consumption, but at a diminishing rate. This finding underscores the importance of optimizing the number of training episodes to balance efficiency gains with the practicality of extended training cycles, providing crucial insights in the future for quicker deployment on the hardware.
The performance of vehicle across different traffic conditions, as portrayed into four distinct regions within the WLTC cycle, is illustrated in Figure (4.6). Each region represents a unique driving environment that significantly impacts vehicle dynamics and thus energy consumption patterns.

- **Urban**: Characterized by slow speeds typical of city traffic conditions, this region often demands by frequent acceleration and deceleration cycles, which can lead to higher energy consumption per unit distance.
- **Suburban**: This region also demands frequent acceleration and deceleration cycles due to traffic conditions but from moderate speeds.
- **Main Road**: With above-medium speeds and low traffic conditions, this region demands few acceleration and deceleration cycles compared to previous two regions.
- **Highway**: High speeds and the absence of traffic conditions demanding the lowest acceleration and deceleration cycles, which can be optimized for lower energy consumption per unit distance.

A derived correlation coefficient of -0.5969 between total energy consumed and vehicle speed indicates a moderately strong inverse relationship. This relationship suggests that higher vehicle speeds typically correlate with reduced total energy consumption, a pattern that is particularly visible in main road and highway conditions. Conversely, the lower speeds characteristic of urban and suburban settings, with their frequent stop-and-go scenarios, correspond to increased energy demands. The average energy consumption across these regions was recorded as 0.7895, 0.7928, 0.7828, and 0.7736 for urban, suburban, main road, and highway regions, respectively. These values underscore an improvement in energy efficiency as vehicular speed increases and as less frequent stopping occurs.
The dashed horizontal line at 0.359 indicates the speed above which vehicle tends to achieve better energy efficiency in the context of actuator utilization in the thermal system. Below this threshold, cooling efficiency reliant on ambient air may be inefficient, and often depend on the radiator fan, which consequently increasing the energy consumption.

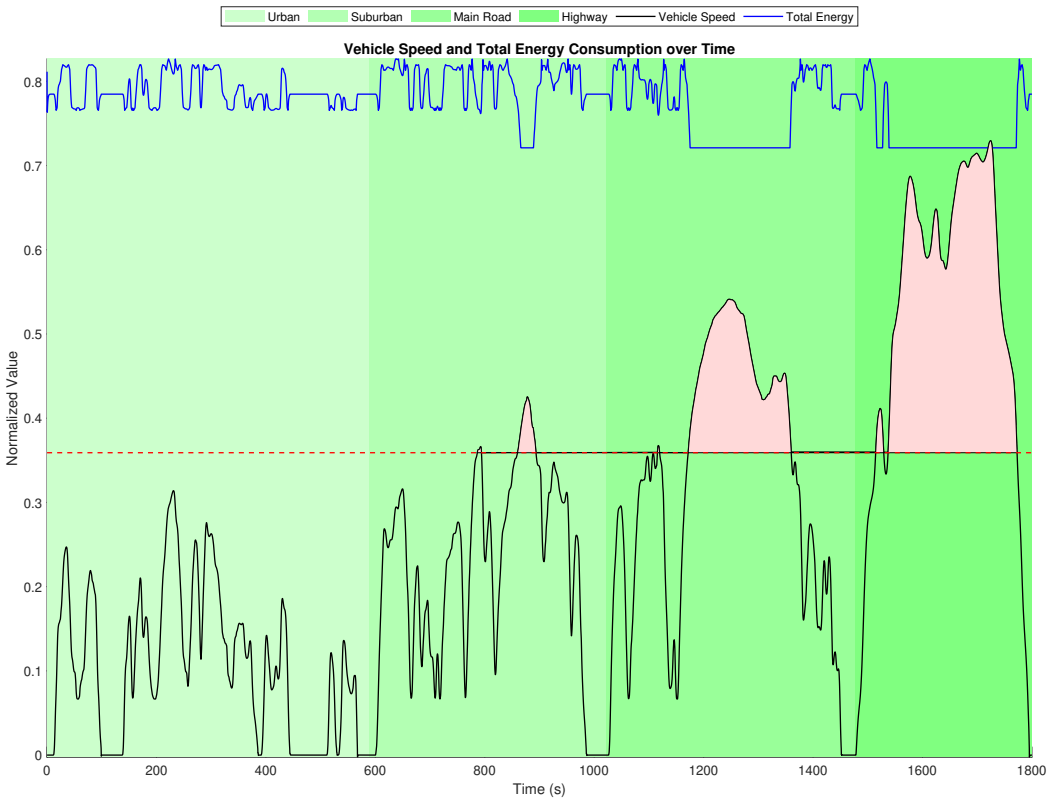


Figure 4.6: Normalized total energy consumption and vehicle speed for different regions in WLTC cycle

5
Future work

The exploration of machine learning techniques in thermal management for BEVs has opened several avenues for further research and development.
A systematic **benchmarking** exercise is crucial to quantitatively assess the impact of the developed cost function on vehicle performance and energy efficiency. This would involve comparative studies to measure the benefits of integrating the cost function, providing concrete evidence to support its wider adoption.
Further refinement of the learning process can be achieved by incorporating a **Gaussian-based policy optimized using the Guided Policy** Search (GPS) as highlighted in the foundational literature of GCL[13], which could significantly enhance policy parameters, leading to superior outcomes in operational stability and energy efficiency. [1] and [14] may be extensively used as starting point while implementing GPS.
Enhancing simulation performance by **compiling the Simulink model** into C-code using tools like Simulink Coder or TargetLink will allow the execution of more complex models in real-time applications.
Additionally, designing a **careful exploration strategy** during the training phase of reinforcement learning is essential to prevent system instability. Smooth transitions in control actions, such as gradual changes in pump speeds, will mitigate risks and enhance system reliability.
Generalizing the cost function to accommodate diverse driving scenarios and conditions is another critical area. This includes adapting the cost function for various initial conditions and driving profiles to ensure broader applicability.
The adoption of Physics Inspired Neural Networks **(PINNs) as a cost function** is a promising direction for future research. PINNs integrate physical principles into the neural network architecture, offering more accurate predictions and potentially reducing the need for extensive data. This approach could lead to a deeper understanding of system behavior, ensuring the cost function optimizes energy efficiency while adhering to essential physical constraints.
Exploring **constraint programming to handle various inequalities** and system constraints could lead to more sophisticated control strategies, considering multiple objectives simultaneously and fostering smarter, adaptive vehicle control systems.
These initiatives will not only enhance technical capabilities but also contribute to the development of more sustainable and efficient systems, pushing forward the boundaries in the field of intelligent transportation systems.

6
Conclusion

This thesis has successfully demonstrated the application of machine learning techniques to develop a parameterized cost function for thermal management in BEVs. Through the integration of IRL, the study has outlined a methodical approach to capture the intricate relationships between various operational parameters of BEVs, thereby enhancing thermal efficiency and vehicle performance.

The use of a machine learning-assisted framework has circumvented the limitations inherent in traditional techniques requiring explicit modeling. This approach has introduced a novel thermal cost predictions technique. Throughout this research, various driving scenarios within the WLTC cycle were analyzed, revealing how different traffic conditions impact energy consumption. It was observed that higher speeds generally correlate with decreased energy usage, underscoring the efficiency of BEVs in less congested environments like highways and main roads. Conversely, urban and suburban settings, characterized by frequent stops and starts, demonstrated higher energy demands, highlighting the challenges and areas for improvement in vehicle system design and urban traffic management.

Future work suggested by this research includes enhancing simulation performance, refining learning processes with advanced algorithms like Guided Policy Search, and exploring innovative strategies such as Physics Inspired Neural Networks. These areas promise to further elevate the adaptability and effectiveness of cost functions in varied driving conditions.

The findings from this thesis contribute significantly to the ongoing efforts to enhance the sustainability and efficiency of BEVs. They lay a solid foundation for future research that can extend these methodologies to broader applications in intelligent transportation systems, ultimately leading to smarter, more energy-efficient vehicle solutions.

Bibliography

[1] C. Finn et al. *Guided Policy Search Code Implementation*. Software available from rll.berkeley.edu/gps. 2016. URL: http://rll.berkeley.edu/gps.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Pearson, 2016.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[4] *CO2 emissions from cars: facts and figures (infographics) | Topics | European Parliament*. en. Mar. 2019. URL: https://www.europarl.europa.eu/topics/en/article/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics (visited on 02/13/2024).

[5] Yuanli Ding et al. "Automotive Li-Ion Batteries: Current Status and Future Perspectives". en. In: *Electrochemical Energy Reviews* 2.1 (Mar. 2019), pp. 1–28. ISSN: 2520-8136. DOI: 10.1007/s41918-018-0022-z. URL: https://doi.org/10.1007/s41918-018-0022-z (visited on 02/14/2024).

[6] Amy Loutf Neziha Akalin. "Reinforcement Learning Approaches in Social Robotics - Scientific Figure on ResearchGate." In: (2020). URL: https://www.researchgate.net/figure/Taxonomy-of-Reinforcement-Learning-algorithms-reproduced-and-shortened-from-37_fig9_344335144.

[7] P. A. Beling S. Adams T. Cody. "A survey of inverse reinforcement learning". In: *Artificial Intelligence Review* (2022). URL: https://doi.org/10.1007/s10462-021-10108-x.

[8] Saman Taheri, Paniz Hosseini, and Ali Razban. "Model predictive control of heating, ventilation, and air conditioning (HVAC) systems: A state-of-the-art review". In: *Journal of Building Engineering* (2022), p. 105067.

[9] Fayez Alanazi. "Electric Vehicles: Benefits, Challenges, and Potential Solutions for Widespread Adaptation". en. In: *Applied Sciences* 13.10 (Jan. 2023), p. 6016. ISSN: 2076-3417. DOI: 10.3390/app13106016. URL: https://www.mdpi.com/2076-3417/13/10/6016 (visited on 02/14/2024).

[10] *A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python*. URL: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/.

[11] Mauro Comi. *Summary: Maximum Entropy Inverse Reinforcement Learning*. URL: https://maurocomi.com/paper_notes/Maximum_Entropy_Inverse_Reinforcement_Learning.pdf.

[12] *Gradient Descent*. URL: https://deepjyotibhattacvharjee.medium.com/gradient-descent-22f50c98be39.

[13] *Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization.* URL: https://arxiv.org/pdf/1603.00448.

[14] *Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics.* URL: https://papers.nips.cc/paper_files/paper/2014/file/6766aa2750c19aad2fa1b32f36ed4aee-Paper.pdf.

[15] *Maximum Entropy Inverse Reinforcement Learning.* URL: https://cdn.aaai.org/AAAI/2008/AAAI08-227.pdf.

[16] *Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning.* URL: https://sites.google.com/view/mbmf.

[17] *Worldwide Harmonized Light Vehicles Test Cycle (WLTC) | Emission Test Cycle | diselnet.* URL: https://dieselnet.com/standards/cycles/wltp.php#cycles.

[18] *Official website for openAI gym environment.* URL: https://www.gymlibrary.dev/content/basic_usage/.

[19] *Machine learning (2024) Wikipedia. Available at:* URL: https://en.wikipedia.org/wiki/Machine_learning.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY