

**FEDERAL UNIVERSITY OF TECHNOLOGY — PARANÁ**

**JOÃO EDUARDO HOFFMANN**

**PROJETO E DESENVOLVIMENTO DE UM SISTEMA  
DE CONTROLE PREDITIVO BASEADO EM MODELO  
PARA GERENCIAMENTO TÉRMICO AUTOMOTIVO**

**THESIS**

**PONTA GROSSA**


**2022**

**JOÃO EDUARDO HOFFMANN** ✉ 

**DESIGN AND DEVELOPMENT OF A MODEL-BASED  
PREDICTIVE CONTROL SYSTEM FOR  
AUTOMOTIVE THERMAL MANAGEMENT**

**Projeto e Desenvolvimento de um Sistema de Controle Preditivo  
Baseado em Modelo para Gerenciamento Térmico Automotivo**

Thesis presented as a requirement to obtain the title of Master in Electrical Engineering at the Federal University of Technology — Paraná (UTFPR).

Advisor: Prof. Ph.D. Max Mauro Dias Santos ✉ 

**PONTA GROSSA**

**2022**



4.0 Internacional

Esta licença permite que outros remixem, adaptem e criem a partir do trabalho para fins não comerciais, desde que atribuem o devido crédito e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.



**Ministério da Educação  
Universidade Tecnológica Federal do Paraná  
Campus Ponta Grossa**



JOAO EDUARDO HOFFMANN

**PROJETO E DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PREDITIVO BASEADO EM MODELO  
PARA GERENCIAMENTO TÉRMICO AUTOMOTIVO**

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Controle E Processamento De Energia.

Data de aprovação: 20 de Dezembro de 2022

Dr. Max Mauro Dias Santos, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Fernanda Cristina Correa, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Roberson Assis De Oliveira, Doutorado - Volvo do Brasil Veículos Ltda.

Dr. Thiago Barros Murari, Doutorado - Faculdade de Tecnologia Senai Cimatec (Senai Cimatec)

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 20/12/2022.

## ACKNOWLEDGMENTS

Agradeço primeiramente a Deus pela minha vida e por ter me dado saúde, força e o conhecimento necessário para finalizar este projeto de pesquisa.

Ao meu pai Jarbas e minha Avó Anna, pelo amor e apoio em todos os passos e decisões envolvidos no projeto de pesquisa.

Ao meu orientador Prof. Dr. Max Mauro Dias Santos, por seu conselho e zelo, por apontar os caminhos a serem seguidos, pelo seu apto suporte e pela confiança depositada.

Aos professores do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa, por todas as contribuições ao projeto de pesquisa e à todos professores que contribuíram na minha trajetória acadêmica.

Ao gerente de projetos Luiz Vicente Balcewicz da Volvo, pelos esforços envolvidos na concepção e manutenção deste projeto, essenciais para que todos os conteúdos descritos neste trabalho se tornassem realidade.

Em especial aos engenheiros Rogério Nalin, Tiago Alexandre Rosso e Henrique Teixeira Avila da Volvo, por me auxiliar na concepção deste projeto desafiador, pelo suporte e tempo dedicado ao projeto, assim como por sua paciência e todos os ensinamentos passados.

A todos os demais que de alguma forma contribuíram para meu crescimento pessoal e profissional.

E por fim, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Apoio à Educação, Pesquisa e Desenvolvimento Científico e Tecnológico (FUNTEF) da UTFPR, Fundação Araucária e Volvo Group, pelo apoio financeiro.

Chamada Pública 02/2020 – Programa de Bolsas Fundação Araucária & Volvo do Brasil Veículos Ltda. Termo de Colaboração 09/2021 (FUNTEF & Fundação Araucária).

## RESUMO

Nos sistemas de gerenciamento térmico automotivo, uma operação de refrigeração adaptativa é necessária pois a rejeição de calor do motor muda constantemente com a dinâmica do veículo. Um objetivo de refrigeração adaptativa ideal para o motor pode ser alcançado ao fornecer um fluxo de líquido refrigerante, em determinados estados de temperatura, o mais próximo possível das necessidades reais de coleta e rejeição de calor. Atualmente, para a regulação da temperatura do líquido refrigerante, sistemas veiculares de controle empregam amplamente controladores proporcional integral derivativo (PID), como processos de controle rápidos e leves, que geram demandas de velocidade do ventilador para o processo de atuação na rejeição de calor. Embora interessante em termos de custo computacional de processamento no veículo, a indução de distúrbios, a partir de parâmetros não estimados com precisão na modelagem do sistema, agrava a regulação da temperatura do líquido refrigerante na presença de diversas variáveis de impacto térmico. Com o objetivo de reduzir tais distúrbios, uma estratégia de controle preditivo baseado em modelo (MPC) é proposta, como um método, algoritmo e estratégia aplicados em sistemas de controle de ventiladores de veículos para a geração de demandas otimizadas de velocidade do ventilador na manutenção de um horizonte previsto de temperaturas do líquido refrigerante para uma configuração de *set point* de temperatura. Melhorias no desempenho do veículo, eficiência de combustível e emissões são potencialmente alcançadas com estratégias de aprendizado de máquina na previsão da temperatura e regulação térmica do líquido refrigerante e dos estados de impacto térmico em um horizonte futuro, definido para permitir que um modelo proposto de rotulagem com aprendizado por reforço (RL) realize buscas por velocidades ideais do ventilador. A estratégia probabilística do agente do modelo de rotulagem é aprimorada na interação e observação da resposta da temperatura do líquido refrigerante, a partir de um modelo de resposta térmica, com confiança em correlações de tempo cruzado com variáveis de impacto térmico, resultando em um menor desvio do *set point* de temperatura, comparado com controladores clássicos. Além disso, é proposto um processo de extração de características interpretáveis por humanos, com o uso do método de agrupamento baseado em covariância inversa Toeplitz (TICC), como um método de extração de estruturas precisas e interpretáveis em dados de séries temporais multivariadas, abordando otimizações no tempo de processamento na aplicação de representações confiáveis e de baixa dimensão. Os resultados de uma avaliação física experimental demonstram a eficácia da solução MPC em comparação com um controlador clássico, ao alcançar as potenciais reduções de 1,53% e 0,61% nos consumos de potência do ventilador e combustível, respectivamente.

**Palavras-chave:** controle preditivo baseado em modelo; aprendizado por reforço; agrupamento baseado em modelo.

## ABSTRACT

In automotive thermal management systems, an adaptive cooling operation is required as the engine's heat rejection is constantly changing with the vehicle dynamics. Maintaining an optimal adaptive engine cooling can be achieved by delivering coolant flow at certain temperature states, as close as possible to actual needs. Currently, for the coolant temperature regulation, cooling fan control systems widely employ Proportional Integral Derivative (PID) controllers, as fast and light solutions in generating fan speed demands for the heat rejection process. Although interesting in terms of in-vehicle computational processing cost, the induction of disturbances from parameters not precisely estimated, in the system modelling, aggravate the low robustness in the regulation of the coolant temperature in the presence of uncertain thermal impacts. Aiming at improving the reduction of disturbances, a Model Predictive Control (MPC) strategy is proposed, as a method, algorithm, and strategy, applied on cooling fan control systems, for the generation of optimized fan speed demands in maintaining a predicted horizon of coolant temperatures at a set point configuration. Improvements on vehicle performance, fuel efficiency and emissions are potentially achieved with the application of machine learning strategies for the prediction and thermal optimization of the coolant temperature in a future control horizon, allowing a proposed Reinforcement Learning (RL) labeling model to perform searches for optimal fan speeds. The probabilistic strategy of the RL agent is improved in interacting and observing the coolant temperature response, from a thermal response model, with confidence from cross-time correlations with thermal impact variables, resulting in less deviance from configurable temperature set points when compared to classic feedback controllers. In addition, a human interpretable feature extraction process is proposed, using the Toeplitz Inverse Covariance-Based Clustering (TICC) method, in extracting accurate and interpretable structures in multivariate time series data, for addressing processing time concerns with the use of reliable and low dimensional feature representations. The results of an experimental physical evaluation demonstrate the effectiveness of the MPC solution in comparison to a classic controller, as it achieves the potential reductions of 1.53% and 0.61% in the consumption of fan power and fuel, respectively.

**Keywords:** model predictive control; reinforcement learning; model-based clustering.

## LIST OF ALGORITHMS

Algorithm 1 – Extract the feature space . . . . .	33
Algorithm 2 – Optimize fan actuation and extract labels . . . . .	44
Algorithm 3 – Acquire the data of interest . . . . .	52
Algorithm 4 – Execute the controller inference . . . . .	52
Algorithm 5 – Impose the fan speed values . . . . .	53
Algorithm 6 – Log the data of interest . . . . .	53

## LIST OF FIGURES

Figure 1 – PID controller architecture for a MIMO system. . . . .	19
Figure 2 – The structure of MPC. . . . .	20
Figure 3 – The structure of adaptive MPC. . . . .	22
Figure 4 – RL perception-action-learning loop. . . . .	26
Figure 5 – Actor-critic setup. . . . .	30
Figure 6 – Human-interpretable feature extraction method. . . . .	32
Figure 7 – Custom and derived DA-RNN architectures. . . . .	35
Figure 8 – Proposed MPC method. . . . .	37
Figure 9 – RL labeling model. . . . .	39
Figure 10 – Visual representations of the action space. . . . .	41
Figure 11 – Internal structure. . . . .	46
Figure 12 – Forecasting architecture. . . . .	47
Figure 13 – Controller architecture. . . . .	48
Figure 14 – Agent architecture. . . . .	49
Figure 15 – Initial action composition. . . . .	50
Figure 16 – Configured ATI VISION screen for visualization. . . . .	54
Figure 17 – Outputs of the TICC visualization algorithm. . . . .	56
Figure 18 – Error $\times$ prediction window curve. . . . .	58
Figure 19 – Accuracy of the target series forecasting and environment models. . . . .	61
Figure 20 – Single-class performance of the driving series forecasting model. . . . .	62
Figure 21 – Reward evolution of the labeling model. . . . .	63
Figure 22 – Results of the MPC solution. . . . .	64
Figure 23 – Results of the extraction of reliable samples. . . . .	66
Figure 24 – Regulation performance of the MPC and PID solutions. . . . .	68



## LIST OF TABLES

Table 1	– Feature extraction evaluation parameters. . . . .	35
Table 2	– Action setting. . . . .	40
Table 3	– MPC evaluation parameters. . . . .	50
Table 4	– Importance matrix. . . . .	56
Table 5	– Evaluation results. . . . .	58
Table 6	– Accuracy of the driving series forecasting model. . . . .	62

## LIST OF ABBREVIATIONS, INITIALS, AND ACRONYMS

### Initials

AE	Autoencoder
API	Application Programming Interface
AUC	Area Under the Curve
BC	Betweenness Centrality
CCP	CAN Calibration Protocol
CNN	Convolutional Neural Network
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
DA-RNN	Dual-Stage Attention-Based Recurrent Neural Network
DDPG	Deep Deterministic Policy Gradients
DL	Deep Learning
DNN	Deep Neural Network
DOA	Domain of Attraction
DPG	Deterministic Policy Gradients
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ECU	Electronic Control Units
FLC	Fuzzy Logic Controller
FUNTEF	Fundação de Apoio à Educação, Pesquisa e Desenvolvimento Científico e Tecnológico
GPS	Global Positioning System
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
LTI	Linear Time-Invariant
MAD	Mean Average Deviation
MAE	Mean Absolute Error
MIMO	Multi-Input Multi-Output
ML	Machine Learning
mp-QP	Multi-Parametric Quadratic Programming
MPC	Model Predictive Control
MSE	Mean Squared Error
NAF	Normalized Advantage Function
NARX	Nonlinear Autoregressive Exogenous
NN	Neural Network
PCA	Principal Component Analysis
PCP	Percentage of Correct Predictions
PID	Proportional Integral Derivative
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARSA	State-Action-Reward-State-Action
SR	Successor Representation
t-SNE	T-Distributed Stochastic Neighbor Embedding
TD	Temporal Difference

TICC      Toeplitz Inverse Covariance-Based Clustering  
UTFPR    Universidade Tecnológica Federal do Paraná

## SUMMARY

<b>1</b>	<b>INTRODUCTION</b>	<b>13</b>
<b>1.1</b>	<b>Control and optimization systems</b>	<b>13</b>
<b>1.2</b>	<b>Feature extraction</b>	<b>14</b>
<b>1.3</b>	<b>Justification</b>	<b>16</b>
<b>1.4</b>	<b>Proposition</b>	<b>17</b>
<b>1.5</b>	<b>Objectives</b>	<b>17</b>
<b>1.6</b>	<b>Contributions</b>	<b>17</b>
<b>1.7</b>	<b>Thesis structure</b>	<b>18</b>
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>19</b>
<b>2.1</b>	<b>Control and Optimization Systems</b>	<b>19</b>
2.1.1	Model Predictive Control	19
2.1.2	MPC formulation	20
2.1.2.1	<u>Highly nonlinear systems and performance concerns</u>	21
2.1.3	Reinforcement Learning	25
2.1.3.1	<u>Reward-Driven Behavior</u>	25
2.1.3.2	<u>Markovian Decision Process</u>	27
2.1.3.3	<u>Challenges in RL</u>	27
2.1.3.4	<u>Value Functions</u>	28
2.1.3.5	<u>Policy Search</u>	29
2.1.3.6	<u>Actor-critic Methods</u>	30
2.1.3.7	<u>Model-based RL</u>	31
<b>3</b>	<b>MATERIAL AND METHODS</b>	<b>32</b>
<b>3.1</b>	<b>Human-interpretable feature extraction</b>	<b>32</b>
3.1.1	Implementation	33
3.1.2	Experimental evaluation	34
3.1.2.1	<u>Models and parameter settings</u>	34
<b>3.2</b>	<b>Model Predictive Controller</b>	<b>35</b>
3.2.1	Coolant temperature regulation architecture	36
3.2.2	Reinforcement learning labeling	38
3.2.2.1	<u>RL labeling interactive setting for coolant temperature regulation</u>	38
3.2.3	In-vehicle model framework and strategy	45
3.2.4	Experimental evaluation	46
3.2.4.1	<u>In-vehicle architecture and parameter settings</u>	46
3.2.4.2	<u>RL labeling architecture and parameter settings</u>	49
3.2.5	Physical evaluation	51
<b>4</b>	<b>RESULTS AND DISCUSSION</b>	<b>55</b>
<b>4.1</b>	<b>Feature extraction</b>	<b>55</b>
4.1.1	Dataset preparation and feature extraction setup	55
4.1.2	Evaluation metrics and structure	57
4.1.3	Evaluation summary	59
<b>4.2</b>	<b>Model predictive controller</b>	<b>60</b>
4.2.1	Dataset preparation	60
4.2.2	Training and testing	61
4.2.3	Physical evaluation	65

<b>5</b>	<b>CONCLUSION</b> . . . . .	<b>69</b>
<b>6</b>	<b>NEXT STEPS</b> . . . . .	<b>70</b>
	<b>REFERENCES</b> . . . . .	<b>71</b>

# 1 INTRODUCTION

## 1.1 Control and optimization systems

Many strategies of dynamic response modeling for control systems have been researched and developed as the task of influencing the disturbance rejection and maintaining a low mathematical complexity level in a controller model design and parameter tuning, as needed, grow apart in high-order system dynamics that carry complex temporal impact correlations. Such characteristics are observed in various vehicle dynamics systems meant for regulation, and in the case of a thermal management strategy, multiple controllers and integration methods can be applied to ensure the optimal regulation of the engine temperature by cooling agents, ensuring the optimal flow and temperature for an refrigerant fluid in various possible thermal response scenarios.

For many decades, PID controllers have been widely applied for the regulation of the coolant temperature at a configurable set point with a determined fan speed. PID controllers present loop mechanisms that employ feedback and are designed with the application of tuning methods on gains, composing control terms that attempt to minimize the error, from a defined set point, over time. Without the model and observing the system response, the controller lacks direct knowledge of the process and, overall, presents a reactive performance that can compromise or poorly accomplish the regulation to a desired set point on certain applications. The performance of the controller can also be impacted by the presence of nonlinearities, resulting in imprecise regulation outputs over changes in uncovered system dynamics behaviors, and the addition of lag in responding to large disturbances.

In covering complex system dynamics in the regulation process while maintaining human interpretability, the Fuzzy Logic Controller (FLC) can be applied with a design that implements a wide range of operating conditions, readily adapted in terms of natural language, allowing a more complex knowledge of a given system's dynamics by eliminating the need to linearize nonlinear systems before tuning controller parameters. In addition, FLC can be flexibly adjusted with new fuzzy rules, exceptions and new behaviors as the base structure is semantic (FENG, 2006). As systems tend to high-order dynamics, the configuration of accurate rules become a difficult task that can be overcome with optimization models in recognizing non-interpretable patterns.

Given the recent advances in the integration of systems' identification and optimization

strategies, the control modeling design for complex systems can be automated by an improved knowledge of the system dynamics from high-order observations, where the regulation can be expressed by cost minimization functions that explore a dynamic environment in the search for disturbance rejection improvements, extending the controller actuation to cover the regulation accuracy of a future horizon in a predictive manner. By employing an explicit model of the controlling plant in predicting future output behavior, the Model Predictive Control (MPC) has the capability of solving nonlinear multivariate control problems on line by searching for control action optimizations in minimizing a specified cost function, with or without constraints (SWIEF; EL-ZAWAWI; EL-HABROUK, 2019). In applying machine and Deep Learning (DL) methods and strategies in system modeling and identification, an improved knowledge over adverse environments can be achieved and its use to reinforce unusual dynamic control behaviors can increase a model's prediction confidence to uncertainty and noise.

## 1.2 Feature extraction

Feature extraction is widely applied in Machine Learning (ML) for classification and prediction with methods that aim to optimally filter redundant and irrelevant information, extracting relevant patterns regarding task context. With the recent increase in the dimensionality of data, feature extraction methods face a proportional challenge in effectively extracting relevant patterns. Similarly, the interpretability of sparse representations becomes harder to achieve as model parameters tend to scale with the dimensionality of inputs, increasing the size of encoded latent spaces.

The Principal Component Analysis (PCA) (JOLLIFFE; CADIMA, 2016) is a classic and statistical technique of feature extraction that adaptively creates a set of lower dimension variables that relate to the variance found in an original distribution. A similar strategy for dimensionality reduction is the Linear Discriminant Analysis (LDA) (THARWAT *et al.*, 2017), which, as a supervised method, maximizes the separability in groups of data that present a similar variation for each specific category. Although simple and fast, PCA suffers from interpretability when capturing the maximum variation in the data and LDA requires a classification that can be difficult to achieve in high-dimensional distributions.

Currently, nonlinear methods, such as t-Distributed Stochastic Neighbor Embedding (t-SNE) (MAATEN; HINTON, 2008) and Autoencoders (AEs) are widely applied to capture nonlinear patterns in high-dimensional data. The t-SNE enables data exploration and visualization

by preserving small pairwise distances instead of large distances, as observed in PCA for the maximization of variance in generated data, capturing nonlinear relationships and improving the representation of high-dimensional data. The AE (CHO; MERRIËNBOER, *et al.*, 2014; SUTSKEVER; VINYALS; LE, 2014) is a Neural Network (NN) architecture used in multiple ML applications. Its internal structure is composed of encoder, latent space, and decoder layers, presenting compression and decompression processes that can assume distinct input and output dimensions and, as part of the training process, produce useful features in intermediate layers by minimizing a reconstruction error in the process. The flexibility in defining sizes for each layer eases the scaling of the structure in regards to the problem, optimally capturing nonlinear patterns in the latent space for high-dimensional inputs.

Advances in ML, such as AEs, have enabled the expansion of problems and solutions in neural machine translation, computer vision, and time series prediction, among others. A wide range of time series prediction algorithms have been proposed in applications regarding signal treatment and analysis (MAKRIDAKIS; HIBON, 1997) and due to the increase in data dimensionality and its nonlinear patterns, Deep Neural Networks (DNNs) have become a research focus in representing high-dimensional and sequential data.

For various real world applications, the impact from exogenous series is essential to accurately describe concurring and future variable states. To address this issue, various Nonlinear Autoregressive Exogenous (NARX) models (LIN *et al.*, 1996; GAO; ER, 2005; DIACONESCU, 2008; YAN; ELGAMAL; COTTRELL, 2013; QIN *et al.*, 2017) have been developed in predicting the values of a given time series using the relationship between the target and driving (exogenous) series. In recent years, Recurrent Neural Networks (RNNs) have been applied as NN structures fit for sequence modeling, flexibly enabling the capture of nonlinear relationships. Further improvements in capturing long-term dependencies were achieved with Long Short-Term Memory (LSTMs) (HOCHREITER; SCHMIDHUBER, 1997) units, which overcome the problem of vanishing gradients (BENGIO; SIMARD; FRASCONI, 1994). For time series AEs, attention mechanisms can be employed to select and propagate features to subsequent layers of an NN model by applying weights that reflect the "importance" of a current hidden state with respect to previous information. In this way, a decoder attention mechanism can relieve the encoding process by filtering information as it most contributes to a task-related goal (CHO; MERRIENBOER, *et al.*, 2014). Using state-of-the-art RNN architectures and attention based AEs, the Dual-Stage Attention-Based RNN (DA-RNN) model enables the adaptive selection of the most relevant features during the prediction process, outperforming classic and state-of-the-art



RNN models for time series prediction (QIN *et al.*, 2017).

Despite accurately capturing relevant patterns, AE latent spaces tend to lose human-level interpretability from high-dimensional inputs, aggravating the prediction accuracy and evaluation from drifts in captured data patterns, described in human-interpretable environments and not covered by overspecified feature compression processes, which are harder to identify and correct as they deviate from human-interpretability (RYBAKOV *et al.*, 2020). To overcome this challenge, the application of clustering techniques for high-dimensional data (HALLAC; NYSTRUP; BOYD, 2016; HIMBERG *et al.*, 2001; BEGUM *et al.*, 2015; SMYTH, 1996; BERNDT; CLIFFORD, 1994) can produce human-interpretable data representations from dynamic programming and optimization functions. General clustering techniques tend to suffer as distance and density based approaches (NA; XUMIN; YONG, 2010; MÜLLNER, 2011; ESTER *et al.*, 1996) are challenged by high data sparsity in high-dimensional spaces, presenting low tolerance to noise and data asymmetry. Furthermore, graph-based approaches (NG; JORDAN; WEISS, 2001; XU; SU, 2015; HALLAC; VARE, *et al.*, 2018) were proposed to allow the expression of cluster relationships by the number and strength of links between and within clusters, improving the estimation of dependency structures in high-dimensional data. For multivariate time series data, dynamic programming and iterative cost minimization functions can optimally support the graphic assignment of clusters for correlated series over a discrete time window, where the definition of parameters leads to human-interpretable data representations, as observed with the Toeplitz Inverse Covariance-Based Clustering (TICC) method (HALLAC; VARE, *et al.*, 2018).

### 1.3 Justification

Although classic control systems present light and fast control processes, their regulation accuracy and complexity are negatively affected proportionally to the dimension of a given dynamic system (EFHEIJ; ALBAGUL; AMMAR ALBRAIKI, 2019). Such is the case for vehicle and thermal dynamics systems, impacted by multiple internal and external factors that present a high-dimensional correlation toward different thermal rejection states for the vehicle's engine. Advances in ML and intelligent systems have shown that system identification, dimensionality reduction and optimizations can be achieved for highly nonlinear systems while establishing high-level settings for a regulation objective (PILLONETTO *et al.*, 2014; SORZANO; VARGAS; MONTANO, 2014; SUN *et al.*, 2019). Additionally, the use of ML and data-driven resources for optimization can ease the cost of low-level controller designs from classic control solutions.

## 1.4 Proposition

Considering advances in ML for time series forecasting, optimization and feature extraction, we propose an MPC system with the aim of improving coolant temperature regulation, where cooling agent optimizations, for the training of an in-vehicle controller, are achieved with a proposed RL labeling system.

In addressing processing time concerns, a feature extraction system is proposed for the reduction of the dimension of inputs while maintaining their sequential relevance.

## 1.5 Objectives

The main objective of this work is the design, development and evaluation of an MPC system proposition for the thermal regulation of the cooling system.

The project targets 3 specific objectives to accomplish this goal:

- Define the methods and strategies for the control and optimization systems;
- Design and develop the ML models and architectures;
- Evaluate the MPC system.

## 1.6 Contributions

The contributions of the work are summarized as follows:

- Proposition of a multivariate time series feature space based on a state-of-the-art clustering technique, presenting balance of model agnostic and specific patterns.
- Presentation of a feature extraction process that enables the simplification of deep RNN architectures by relieving attention-based expenses.
- Proposition of a predictive cooling system control method, able to reduce coolant temperature disturbances, from thermal impact variables and parameters not precisely estimated on the system modeling, with a horizon of fan speeds that aim to maintain the coolant at a desirable temperature set point.
- Proposition of high-level controller design and offline optimization process, relying on the correct definition of optimization objectives and constraints to set the configuration of a reinforcement learning labeling model.

## 1.7 Thesis structure

The remainder of this work is structured as follows:

Chapter 2 covers the TICC model-based clustering method and the DA-RNN as a NARX model architecture for the literature review of the proposed feature extraction strategy. In addition, MPC and RL are reviewed with recent contributions in highlighting the proposed MPC control strategy.

Chapter 3 describes the proposed feature extraction and MPC control strategies, methodologies and algorithms.

Chapter 4 contains the experimental comparative evaluation, describing the contrast between the prediction accuracy and the number of parameters in the adoption of the proposed feature extraction method.

Chapter 5 presents the conclusion. Finally, the next steps of the project are presented in Chapter 6.

## 2 LITERATURE REVIEW

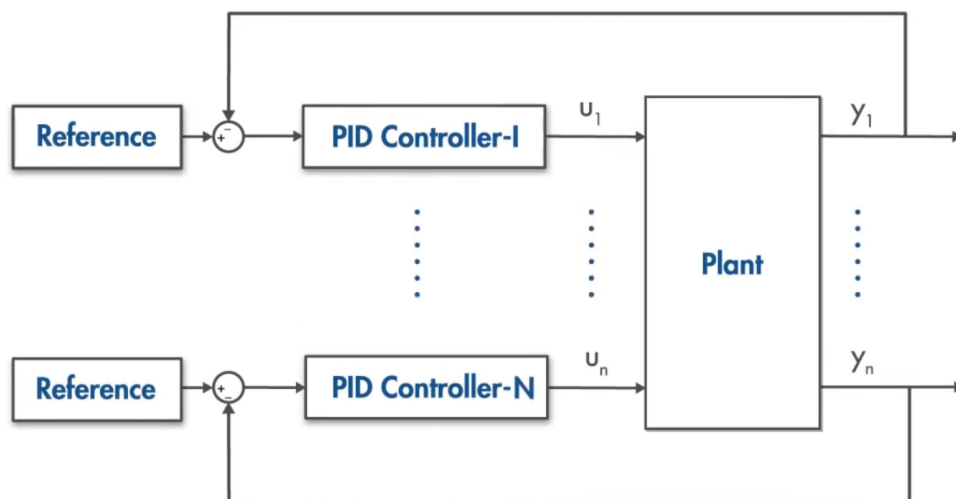
In this chapter, the literature will be reviewed exploring in depth each theme brought on this work. Concepts and definitions around the subjects are presented in this section, alongside the context on where each one is involved and needed. Furthermore, an overview of the applications on the matter is shown, resuming the importance and deliberate approaches found in scientific literature.

### 2.1 Control and Optimization Systems

#### 2.1.1 Model Predictive Control

Model Predictive Control (MPC) is a feedback control system that employs the knowledge of a given system's response to make predictions about the future outputs of a process. With the goal of regulating a plant's response to achieve a certain set point objective, MPC uses the plant's model as key in improving each regulation step based on the future response to a series of controllable inputs. Additionally, MPC handles Multi-Input Multi-Output (MIMO) systems that might present correlations between their inputs and outputs, which is challenging for Proportional Integral Derivative (PID) controllers, given that each control loop operates independently, in a way that the number of controller gains escalate with the number of input and output pairs (SALEM; MOSAAD, 2015). Figure 1 shows the complexity of a PID controller design for MIMO systems that present  $n$  plant inputs  $u$  and outputs  $y$ .

Figure 1 – PID controller architecture for a MIMO system.



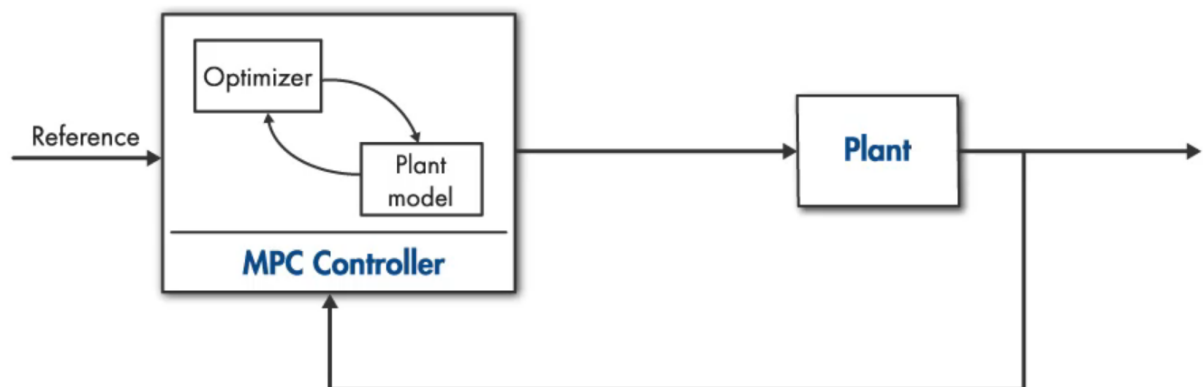
Source: Bemporad, Ricker, and Morari (2020).

MPC offers support for system constraints, aiding the optimization problem when adopting outputs that lead to undesirable consequences, which may affect the safety of the system and other components involved in its response. The limits of a given system can be further incorporated in the form of future references, supporting the accuracy of the regulation task in evolving environments (SALEM; MOSAAD, 2015). In simulating the output trajectory, the MPC controller uses a given system's model and with an optimizer ensures that the predicted future plant output tracks the desired reference. With the definition of a controller prediction horizon  $p$ , the optimization covers the minimization of a weighted cost function  $J$ . The optimization is performed online and at each regulation step (SALEM; MOSAAD, 2015), involving the adoption of  $p$  states for all future actions. The general cost function is expressed by the following equation:

$$J = \sum_{i=1}^p w_e e_{k+i}^2 + \sum_{i=0}^{p-1} w_{\Delta u} \Delta u_{k+i}^2 \quad (1)$$

Here, the collection of errors  $e$  is weighted  $w_e$  in fitting the prediction window for the best future horizon response of the system. Along with the sum of weighted horizon errors for controller inputs starting at  $t = k$ , the collection and addition of weighted actions  $w_{\delta u} \delta u$  compose the cost function in correlating the response of previous step responses in the future performance of the optimizer. The described optimization system often eases covering a great range of MIMO systems with great accuracy at the cost of expensive computational requirements (SALEM; MOSAAD, 2015). Figure 2 presents the MPC.

Figure 2 – The structure of MPC.



Source: Bemporad, Ricker, and Morari (2020).

### 2.1.2 MPC formulation

Considering an uncertain Linear Time-Invariant (LTI) system:

$$x(k+1) = A(\theta^*)x(k) + B(\theta^*)u(k) \quad (2)$$

where  $\theta^* \in \Theta$  is the uncertain parameters in the model for a convex set  $\Theta$ . For a known parameter  $\theta^*$ , the controller inputs over the prediction horizon  $\mathbf{u} = \{u(0), u(1), \dots, u(N-1)\}$  lead to the solution of the following minimization problem:

$$\min_{\mathbf{u}} J_N(x, \mathbf{u}) = \sum_{i=0}^{N-1} l(x(i), u(i)) + V(x(N)) \quad (3)$$

subject to

$$\begin{aligned} x(i+1) &= A(\theta^*)x(i) + B(\theta^*)u(i), x(0) = x \\ x(i) &\in \mathcal{X}, \quad u(i) \in \mathcal{U}, \quad i \in [0, N-1] \\ x(N) &\in \mathcal{X}_f \in \mathcal{X} \end{aligned} \quad (4)$$

where the sets  $\mathcal{U}$  and  $\mathcal{X}$  represent the input and state constraints,  $x(N) \in \mathcal{X}_f$  is the artificial terminal constraint in ensuring stability, and  $V(x(N))$  represents the terminal cost function. In summary, the MPC loss function can be described by:

$$l(x(i), u(i)) = x(i)^T Q x(i) + u(i)^T R u(i) \quad (5)$$

Here, with  $Q$  and  $R$  as positive definite,  $V(x)$  and  $\mathcal{X}_f$  are defined by the control Lyapunov function of  $V(x)$  in  $\mathcal{X}_f$  (KIM, J.-S., 2010), yielding the optimal control sequence:

$$\mathbf{u}^*(x) = \{u^*(0; x), u^*(1; x), \dots, u^*(N-1; x)\} \quad (6)$$

the optimal trajectory:

$$\mathbf{x}^*(x) = \{x^*(0; x), x^*(1; x), \dots, x^*(N; x)\} \quad (7)$$

and the optimal cost:

$$J_N^*(x) = J_N(x, \mathbf{u}^*(x)) \quad (8)$$

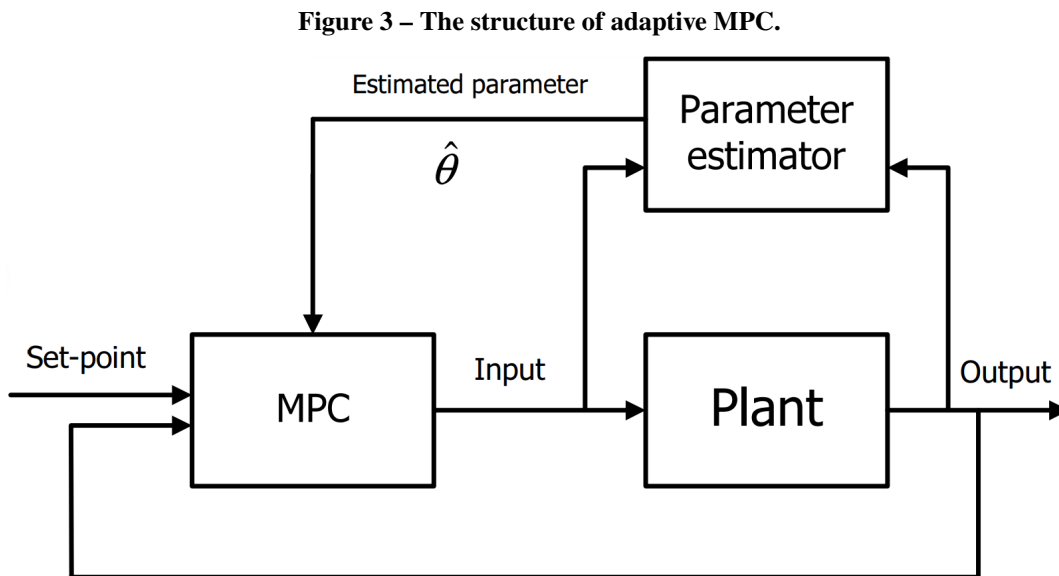
### 2.1.2.1 Highly nonlinear systems and performance concerns

In adapting MPC to real-time applications and complex nonlinear environments, alternatives can be adopted for the optimization model. Such is the Adaptive MPC model, which establishes a plant model that adapts the same number of states and constraints across different nonlinear operating conditions with the intention of performing MPC cost function

minimizations for a linear representation of the regulation function. In this sense, considering the same LTI system as in 2 and that the parameter  $\theta^*$  is now unknown and varies in multiple operating conditions, an estimation can be made with measurements from the plant (KIM, J.-S., 2010). As in popular strategies of employing estimators (HAYKIN, 2002; MIURA, 2011), the prediction equation can then be described by:

$$x(i+1) = A(\hat{\theta})x(i) + B(\hat{\theta})u(i), x(0) = x \quad (9)$$

where  $\hat{\theta}$  denotes the estimated parameter. Figure 3 shows the adaptive MPC structure with the use of an estimator.



Source: Jung-Su Kim (2010).

Many adaptive MPC approaches were developed in the search for optimal closed loop stability (KIM, T.-H.; SUGIE, 2008; SHOUCHE *et al.*, 1998; ADETOLA; DEHAAN; GUAY, 2009; KIM, J.-S.; YOON; SHIM, 2005) and although ensuring that the control process is established over dynamically evolving models, the estimated parameter  $\theta_k$ , from previous measurements  $x(k-1)$ , may not correlate to  $x(k)$ , found in the Domain of Attraction (DOA) of  $(A(\theta_k)x(i), B(\theta_k))$ , leading to infeasibility in multiple applications (KIM, J.-S., 2010).

Similarly, gain-scheduled MPC enables the linearization of a plant with multiple plant models, each representing a distinct state with the advantage of adopting unique constraints, that are evolving in known nonlinear operating conditions. For the execution of the correct plant function, a switching algorithm is adopted for gain-scheduled MPC in understanding a current state operating condition and adopting its defined optimization function. In a gain-scheduled

MPC algorithm, at each sample time  $t$ , given a partitioned state  $x(t) = [x'_1(t), x'_2(t)]'$ , the system can be described as:

$$x(t+1) = A(x_1(t))x(t) + B(x_1(t))v(t) \quad (10)$$

where,

$$u(t) = g(x(t), v(t)) \quad (11)$$

is the precompensation feedback and the control signal  $v(t)$  is obtained from measurement associations with feedback gains  $F_i$  that ensure exponential stability for the nonlinear system around the origin, as in the following equation:

$$v(t) = F_{i(t)}x(t) \quad (12)$$

For the definition of the MPC receding-horizon operation, an auxiliary open-loop component  $c$  is used to complement the control signal and enable the selection of the optimal sequence  $c(t)$ , as in:

$$\mathbf{c}(t) = \arg \min_{\mathbf{c}(t)} \|\mathbf{c}(t)\|^2 \text{ subject to } \begin{bmatrix} x(t) \\ \mathbf{c}(t) \end{bmatrix} \in S_N^{i(t)} \quad (13)$$

where  $i(t) = i(x_1(t))$  is the index, such that  $x_1(t) \in X_i$ , and  $S_N$  is the set of states steered offline by the new control sequence. The new control signal is expressed as:

$$v(t) = F_{i(t)}x(t) + c(t | t) \quad (14)$$

where  $c'(t) = [c'(t | t), c'(t+1 | t), \dots, c'(t+N-1 | t)]$ .

The optimal sequence, at time  $t$  and among all admissible sequences  $c(t)$ , is then selected by the minimum  $l_2$  norm from applying the control signal  $v(t)$  to the plant (CHISCI; FALUGI; ZAPPA, 2003). Although an interesting solution for simpler nonlinear systems, in the case of highly nonlinear systems, the solution of a nonconvex optimization problem, comprising a nonlinear system, its constraints and cost function, is hardly identifiable and thus the application of linearization techniques is mostly ineffective. For this purpose, a nonlinear solution, using regression based approaches for system identification, may be applied (BEMPORAD; RICKER; MORARI, 2020).

In addressing the reduction of computational costs in processing an MPC solution, a



model order reduction strategy may be adopted in removing state variables that contribute less to the plant response. Similar speed and memory usage gains may be obtained with the use of various strategies, such as shortening the prediction and control horizon, reducing the number of constraints and increasing the accepted error in the optimization process, which at the cost of lower precision operations and data representations, decrease computational costs (BEMPORAD; RICKER; MORARI, 2020). Another solution for decreasing the processing time of the controller is the explicit MPC method, which employs an offline optimization solution for all the states within a given prediction range with Multi-Parametric Quadratic Programming (mp-QP). This can be described by rewriting Equation 3 as the convex Quadratic Program (QP):

$$J^*(x(t)) = \frac{1}{2}x^T(t)Gx(t) + \min_{\mathbf{u}} \left\{ \frac{1}{2}\mathbf{u}^T H\mathbf{u} + x^T(t)F\mathbf{u} \right\} \quad (15)$$

subject to

$$A_c\mathbf{u} \leq b_0 + B_c x(t) \quad (16)$$

where  $n \triangleq n_u N$ ,  $n_u$  is the number of controller output variables,  $m$  is the number of state variables,  $q$  is the number of linear inequality constraints imposed in the MPC problem formulation, as in equation 3,  $H \in \mathbb{R}^{n \times n}$  is the Hessian matrix,  $F \in \mathbb{R}^{n \times m}$  defines the linear term,  $G \in \mathbb{R}^{m \times m}$  affects the optimal cost function, and the matrices  $A_c \in \mathbb{R}^{q \times n}$ ,  $b_0 \in \mathbb{R}^q$ , and  $B_c \in \mathbb{R}^{q \times m}$  define a compact form of the constraints, which are computed offline. The MPC law is defined by rewriting the controller manipulated variables as:

$$u(x) = [I \ 0 \ \dots \ 0]z(x) \quad (17)$$

where  $z$  represents the optimizer of the QP problem and  $I$  is the identity matrix of dimension  $n_u \times n_u$  (BEMPORAD, 2013; LEE; CHANG, 2017).

Considering that the optimizer function  $z^* : X_f \mapsto \mathbb{R}^n$  is piecewise affine and continuous over the set  $X_f$  of parameters  $x$  (BEMPORAD; MORARI, *et al.*, 2002), the online computations can be reduced to the offline evaluation of:

$$u(x) = \begin{cases} F_1x + g_1 & \text{if } H_1x \leq i_1 \\ \vdots & \vdots \\ F_Mx + g_M & \text{if } H_Mx \leq i_M \end{cases} \quad (18)$$

In this way, the iterative optimization process is simplified to a linear function evaluation with identification of the state in which the response model is found based on the currently

imposed constraints. As a consequence, the complexity of a given system's regions and its identification algorithm may require a higher memory capacity. To improve memory usage, regions can be merged at the cost of increasing the distance to an optimal solution (BEMPORAD; RICKER; MORARI, 2020).

### 2.1.3 Reinforcement Learning

Given recent advances in machine learning for system identification and its need to describe and control highly nonlinear, multivariate and dynamic environments for the purpose of searching for optimal cost paths while respecting constraints that increase with a given system's complexity. RL comprises a solution in a learning environment that makes use of a high-level view of a given problem in establishing the optimal probabilistic strategy for a given agent or controller. In this sense, an RL system comprises a general optimization problem that evolves in accuracy and may evolve in speed as the knowledge of a given system's response to input state variables for a well established goal is learned iteratively. In increasing the ability to comprise relevant information over a complex system for a specific objective, DNN architectures may efficiently enable the representation of high-dimensional data with structures fit to correlate and process categorical and numerical data.

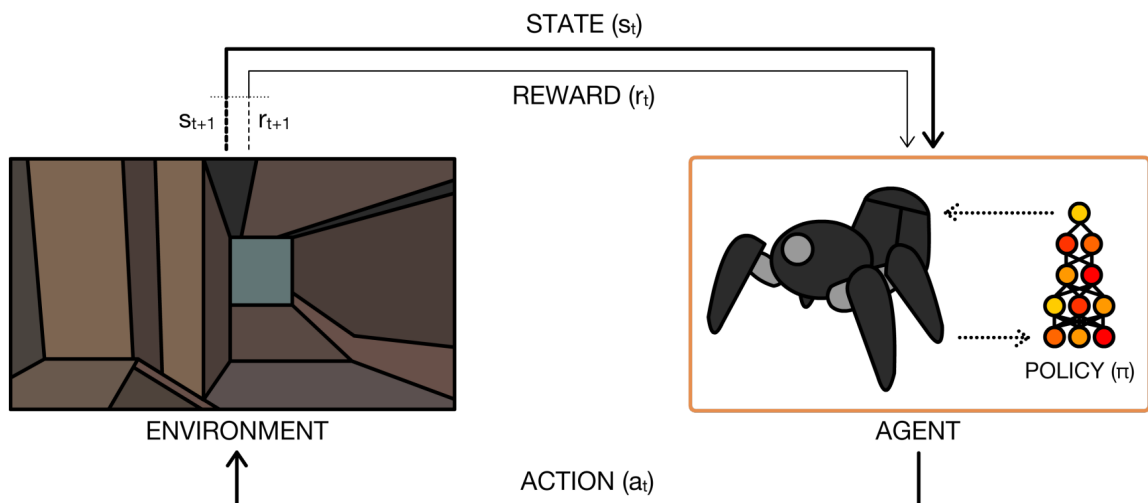
#### 2.1.3.1 Reward-Driven Behavior

An RL system contains resources that enable the learning process through an extensive interaction with a given environment, which simulates the system's response to related inputs. Thus, upon observing the consequences of its actions, an agent can learn to adopt new and improved strategies in reaching a defined goal within this environment. Similar to the inspirations of ML and DL in human behavior, driven by learning the effects of various sensed inputs to understand and take advantage of either learned or instinctive actions for a specific goal in a given environment, RL has its main foundations in behaviorist psychology as the paradigm of trial-and-error. Another RL inspiration is in optimal control, as its efficiency is correlated to dynamic programming (ARULKUMARAN *et al.*, 2017).

In an RL system, a machine learning controlled agent observes an initial state  $s_t$  at the first iteration  $t$  and by interacting with the environment with subsequent actions, starting with action  $a_t$  for state  $s_t$ , either the environment or the agent or both transition to new states

$s_{t+1}$ . Moving forward, and considering that an environment state comprises all the required information of the agent and environment iterative evolution, the agent probabilistic strategy is reinforced as a proper guide, to an objective in the environment, determines if an action or sequence of actions lead to optimal consequences. This feedback system dictates which behavior should be adopted and learned by the agent, as motivation to achieve a reward, provided, as  $r_{t+1}$  for the first environment response, in environment transitions by a system that is designed according to the complexity and dimension of the agent, environment and objectives. This ongoing interaction between agent and environment has the overall goal of constructing a policy  $\pi$ , as a control strategy, that maximizes the expected return. In this sense and unlike other control systems, the controller, as an RL agent, learns to minimize a given cost function by learning its consequences in the environment by trial and error, without the need for model assumptions as in optimal control solutions. In high-dimensional and probabilistic environments, where the knowledge of the plant is scarce, RL employs a system's response learning process to a specific objective within an environment using properties of ML and DL to automatically find compact and relevant low-dimensional representations of inputs, consistently providing new and optimized cost paths to complex systems (ARULKUMARAN *et al.*, 2017). Figure 4 presents the RL interactive-learning loop, where the probabilistic strategy of the agent is reinforced with knowledge from state transitions of the form  $(s_t, a_t, s_{t+1}, r_{t+1})$ .

**Figure 4 – RL perception-action-learning loop.**



Source: Arulkumaran *et al.* (2017).

### 2.1.3.2 Markovian Decision Process

RL can be described as a system with Markov properties, as it contains a set of states  $\mathcal{S}$ , distribution of starting states  $p(s_0)$ , set of actions  $\mathcal{A}$ , transition dynamics  $\mathcal{T}(s_{t+1}|s_t, a_t)$ , which map state-actions to its subsequent distribution of states, and a discount factor  $\gamma \in [0, 1]$ , employed to emphasize immediate rewards. An RL policy maps states to a probability distribution over actions  $\pi : \mathcal{S} \rightarrow p(\mathcal{A} = \mathbf{a} | \mathcal{S})$ , and, in achieving the maximum expected return from all states the optimal policy  $\pi^*$ :

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[R | \pi] \quad (19)$$

In accessing correlations beyond a past state  $s_{t-1}$  to improve the agent's learning capabilities of a given environment, DL solutions may be applied as in RNNs (WIERSTRA *et al.*, 2010; HAUSKNECHT; STONE, 2015; HEESS *et al.*, 2015; MNIH; BADIA; MIRZA; GRAVES; LILLICRAP, T. P., *et al.*, 2016; OH *et al.*, 2016), which are fit for sequential data in aggregating an improved dynamic state to the agent (ARULKUMARAN *et al.*, 2017).

### 2.1.3.3 Challenges in RL

In the field of control systems, several challenges are faced by RL algorithms. The trial-and-error interaction may be computationally extensive and expensive and due to the complexity of a given environment and difficult task of designing a reward system, lead to undesirable policies. Strong temporal correlations need to be addressed as they may yield relevant dynamic characteristics of interactions for a given objective, and long-range time dependencies have consequences as previous actions may lead to relevant consequences after many transitions of the environment (ARULKUMARAN *et al.*, 2017).

In addressing the concerns, strategies can be adopted in estimating the expected return  $\mathbb{E}[R | \theta]$  of being in a given state, as with value functions, or encoding policies with weight optimization to maximize this expected return, as with policy search methods. Both strategies can also be used in a hybrid, actor-critic approach (ARULKUMARAN *et al.*, 2017).

### 2.1.3.4 Value Functions

Value Function methods use a state-value function  $V^*(s)$  that, for a state-value corresponding to the optimal policy  $\pi^*$ , can be defined as:

$$V^\pi(s) = \mathbb{E}[R \mid s, \pi] \quad (20)$$

The maximization of the expected return can be observed as an optimal value function  $V^*(s)$ , which leads to the retrieval of the optimal policy in choosing the action  $a$ , maximizes  $\mathbb{E}_{\mathbf{s}_{t+1} \sim \mathcal{T}(\mathbf{s}_t, \mathbf{a})} [V^*(\mathbf{s}_{t+1})]$  among all actions at  $s_t$ . Considering that, for the RL setting, transition dynamics  $\mathcal{T}$  are unavailable, a quality function  $Q$ , similar to  $V^\pi$ , can be used, as with the initial action  $a$  provided,  $\pi$  is only followed from the subsequent state onward:

$$Q^\pi(s, a) = \mathbb{E}[R \mid s, a, \pi] \quad (21)$$

Here, the best policy is found by greedily choosing an action  $a$  at every state  $s$ , where, under this policy,  $V^\pi(s)$  is found by maximizing  $Q^\pi(s, a)$ . Considering that each quality state  $Q^\pi(s, a)$  aggregates importance toward the given goal, the Markov property can be adopted for current values in improving the  $Q^\pi$  estimate:

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q^\pi(\mathbf{s}_t, \mathbf{a}_t) + \alpha \delta \quad (22)$$

where  $\alpha$  is the learning rate and  $\delta = Y - Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$  represents the Temporal Difference (TD) error, and, as in the standard regression problem,  $Y$  is the target. Here, transitions generated by this derived policy can be used to improve the estimate of  $Q^\pi$ , which results in setting  $Y = r_t + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$ , this on-policy learning algorithm is known as State-Action-Reward-State-Action (SARSA). When updating the estimate with transitions, not necessarily generated by a generated policy, the target becomes  $Y = r_t + \gamma \max_{\mathbf{a}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a})$ , directly approximating  $Q^\pi$ , also known as the Q-learning off-policy algorithm (ARULKUMARAN *et al.*, 2017). As the combinations of states and actions become large, the memory and computation requirements for Q functions proportionally increase. To address this problem, Deep Q-Networks (DQNs) were proposed (MNIH; KAVUKCUOGLU, *et al.*, 2015), initially as a strategy to downsample high-dimensional data with Convolutional NNs (CNNs) and further enabling the use of DNN techniques in approximating  $Q(s, a)$ . A DQN's evaluation network uses  $Q(s, a; \theta)$  as the Q-function for approximating the action value function while a target network uses  $Q(s, a; \theta^-)$ . As

the parameters of the evaluation network are updated, a copy is made to the target network at every  $i - th$  iteration as in the following loss function:

$$L_i(\theta_i) = E_{(s,a,r,s')} \left[ \left( y_i^{\text{DQN}} - Q(s, a; \theta_i) \right)^2 \right], \quad (23)$$

$$y_i^{\text{DQN}} = r + \gamma \max_{a'} Q(s', a'; \theta_i^-),$$

where,  $i$  is the current iteration,  $\theta_i$  are the parameters in the evaluation network and  $\theta_i^-$  are the copied parameters in the target network.

In simplifying the learning of action  $a$  relevance, Advantage functions  $\mathcal{A}^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$  represent relative state-values that are more intuitive for weight optimization, as learning the more exclusive impact of actions for the overall goal is easier than learning the complete return from an action (ARULKUMARAN *et al.*, 2017). Recently, many advantage-based update models have been used in Deep RL (DRL) algorithms (WANG; SCHAUL, *et al.*, 2015; GU, S.; LILLICRAP, T.; SUTSKEVER, *et al.*, 2016; MNIH; BADIA; MIRZA; GRAVES; LILLICRAP, T., *et al.*, 2016; SCHULMAN *et al.*, 2016).

### 2.1.3.5 Policy Search

Unlike value functions, in policy search methods, a parameterised policy  $\pi_\theta$  is updated to maximize the expected return  $\mathbb{E}[R | \theta]$  with gradient-based or gradient-free optimization strategies. Gradients provide support to improve a parameterised policy as an average over acceptable trajectories, observed in the current policy parameterisation, which can be obtained by applying linearization or stochastic approximations. In model-free RL methods, which learn directly from interacting with the environment, the estimate of the expected return from a state can be performed by averaging the return from multiple rollouts of a policy with the Monte Carlo method (SUTTON; BARTO, 2018). In the application of gradient-based learning, a gradient estimator is used, similar to the optimization of the log-likelihood in supervised learning, which increases the relevance of a sampled action weighted by the return. This reinforcing rule computes the gradient of a given expectation over a function  $f$  of a random variable  $X$  in relation to the policy parameters  $\theta$ , as in the following equation:

$$\nabla_\theta \mathbb{E}_X[f(X; \theta)] = \mathbb{E}_X[f(X; \theta) \nabla_\theta \log p(X)] \quad (24)$$

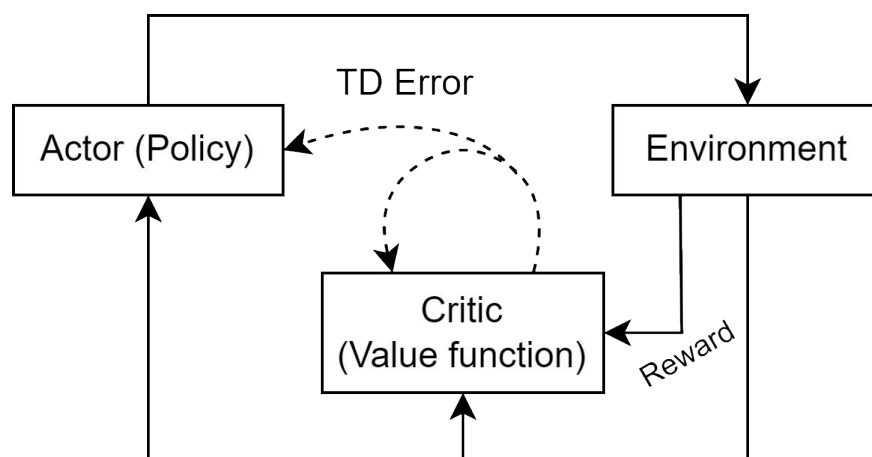
The described computation is generally performed by subtracting a baseline to remove

noise from high variance gradients, as a consequence of relying on the empirical return of a trajectory, establishing weighted updates by an advantage, similar to advantage representations in the description of value functions, rather than the complete return of an action. This strategy can be adopted with an average return over several episodes but may vary according to gains in the problematic (ARULKUMARAN *et al.*, 2017).

### 2.1.3.6 Actor-critic Methods

In combining value functions with policy search strategies, actor-critic methods establish a policy agent that learns through feedback from a value function, also known as critic. Considering that policy search methods use unbiased estimates to reduce gradient variance, with consequent noise, actor-critic methods differ by using an optimized value function. Additionally, actor-critic algorithms can make use of Deterministic Policy Gradients (DPGs), which extend the standard stochastic policies as deterministic. For an environment that does not yield uncertainties for the estimation of actions, simpler policies that integrate over state spaces can be adopted (ARULKUMARAN *et al.*, 2017). This is the case for high-dimensional continuous control problems (SILVER *et al.*, 2014). For high-dimensional state spaces, the processing and description of estimates can benefit from DNNs, thus, NNs can be applied, as function approximators (LILLICRAP, T. P. *et al.*, 2016), in Deep DPGs ((DDPGs)). Figure 5 shows the actor's strategy in choosing actions over environmental states for actor-critic methods. The critic, as the value function, uses the TD error as an outcome of the processing state and reward from a previous interaction, to update itself and the actor.

**Figure 5 – Actor-critic setup.**



Source: Adapted from Arulkumaran *et al.* (2017).

As a benefit of value functions, actor-critic methods can present improved data efficiency from off-policy methods or improved stability from on-policy methods (ARULKUMARAN *et al.*, 2017). In this sense, several attempts were made to integrate or merge both methods (WANG; BAPST, *et al.*, 2017; O'DONOGHUE *et al.*, 2017; GU, S.; LILLICRAP, T.; GHAHRAMANI, *et al.*, 2016; GRUSLYS *et al.*, 2018; GU, S. ( *et al.*, 2017).

#### 2.1.3.7 Model-based RL

Recent developments in system identification enabled the efficient incorporation of complex systems in RL models. Simulation environments, learned as predictive models of dynamical systems, can be embedded into low-dimensional spaces from higher-dimensional observations. Thus, DRL allows the interaction between agents and simple representations of high-dimensional spaces such as images from a camera, which can be later scaled up to high-dimensional visual domains. Given that a system can present high complexity in its response, from nonlinearities, as multivariate and temporal correlations in time series prediction problems, a great number of observations to optimize the simulation model weights may be needed. In this sense, Gu et al. (GU, S.; LILLICRAP, T.; SUTSKEVER, *et al.*, 2016) proposed a strategy of locally training linear models for use with the Normalized Advantage Function (NAF) algorithm, as the continuous equivalent of the DQN (MNIH; KAVUKCUOGLU, *et al.*, 2015). Less common and as a potential method for improving data efficiency, the Successor Representation (SR), can be used to replace the transition dynamics  $\mathcal{T}$  with expected future occupancies, which, when linearly combined with the reward function  $\mathcal{R}$ , are more robust than model-free methods, while failing with  $\mathcal{T}$  changes (ARULKUMARAN *et al.*, 2017). The extension of SRs into DNNs can be used in balancing the data efficiency with the accuracy of probabilistic strategy updates within a complex environment (KULKARNI *et al.*, 2016).



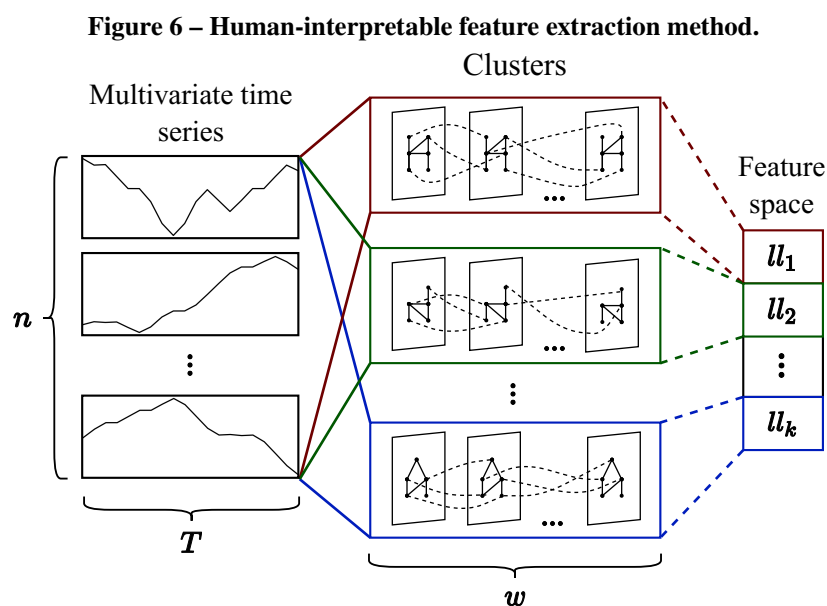
### 3 MATERIAL AND METHODS

In this chapter, all proposed methods, strategies and algorithms will be described, with examples and explanations. Furthermore, methodologies employed and models implemented for the evaluation of the human-interpretable feature extraction method and for the composition of the model-based predictive controller will be presented, highlighting the use of experimental clustering environments in increasing the confidence of comparisons in performance between widely employed and proposed strategies. All the algorithms were implemented in Python, using the TensorFlow Keras API.

According to the complete development of each strategy, the feature extraction method and results are described as previous to the MPC system due to its inspirations, for the design and development, in the architectures that are described in Subsections 3.1.2 and 3.1.2.1.

#### 3.1 Human-interpretable feature extraction

The proposed human-interpretable feature space is composed of the likelihood a given multivariate and sequential sample has of belonging to a given cluster in a human-interpretable environment. The clustering of a multivariate time series  $n \times T$  is performed using the TICC method with a dynamic parameter definition to obtain  $K$  clusters, described by a correlation window of size  $w$  (HALLAC; VARE, *et al.*, 2018). Figure 6 presents the proposed feature extraction method.



Source: Own Authorship (2022).

Given the dynamic definition of TICC (HALLAC; VARE, *et al.*, 2018) parameters while interpreting each cluster labeled sample, after iterating through the overall optimization function  $\mathcal{L}$ , extracted features can be computed as the likelihood a sample has of belonging to each cluster. The proposed feature extraction algorithm iterates through each cluster's optimized information, consisting of the empirical mean  $\mu_i$  and inverse covariance matrix  $\Theta_i$  to obtain the  $i$  dimensional feature space of a given time series sample  $X$ . Algorithm 1 presents the iterative computation of sample likelihoods.

---

**Algorithm 1 – Extract the feature space**

---

**Given**  $X$  = time series of  $T$  sequential observations,  $\mu_i = i$ -th cluster's empirical mean,  $\Theta_i = i$ -th cluster's inverse covariance matrix.  
**Initialize** LL = list of  $K$  zeros.  
1 **for**  $i = 1, \dots, K$  **do**  
2      $x = X - \mu_i$   
3     LD =  $\log(\det(\Theta_i))$   
4     LL[ $i$ ] =  $x^T \odot (\Theta_i \odot x) + LD$   
5 **end for**  
6 **return** LL

---

**Source: Own Authorship (2022).**

---

The feature space  $(\ell_1, \ell_2, \dots, \ell_K)$ , produced at the end of the feature extraction process, has a reduced dimension, comprising multivariate and temporal correlations for a defined window size  $w$ . Thus, in maintaining the importance of sequential information, an original input dimension of  $n \times T$  can be reduced to  $K \times W$ , obtained through multiple iterations of feature extraction, where  $k$  is the set of cluster-related likelihoods and  $W$  is a reduced window of size  $T-w+1$ , if  $T \geq w$ . Starting at  $t = w$ , a static feature space comprises all temporal and multivariate correlations for past states. With the collection of subsequent states up to  $t = W$ , the feature representation expands into a dynamic and changing clustered environment.

### 3.1.1 Implementation

The EM optimization algorithm consists of the combined dynamic programming and ADMM method, where each cluster is initialized randomly, convergence is achieved by alternating between cluster assignments and parameter updates, and interruptions are performed when cluster assignments are stationary. The optimization parameters, consisting of window size  $w$ , number of clusters  $K$ , sparsity level  $\lambda$  and smoothness penalty  $\beta$ , are dynamically defined through the search for human-interpretability in visualizing each cluster's assigned samples and related individual time series values and variability as its weights.

Initially,  $\lambda$  and  $\beta$  are fixed while  $w$  and  $K$  are modified with the goal of visualizing homogeneity with reasonable variance in the quantity of samples for all clusters, showing consistency in iterating for convergence. After establishing a reasonable  $w$  and  $K$ ,  $\lambda$  is modified to improve the cluster assignments for a human-interpretable environment and  $\beta$  in smoothing the cluster assignment switches. For an optimal clustered environment small changes in each individual parameter can be applied. The EM optimization and feature extraction algorithms were built and run in Python.

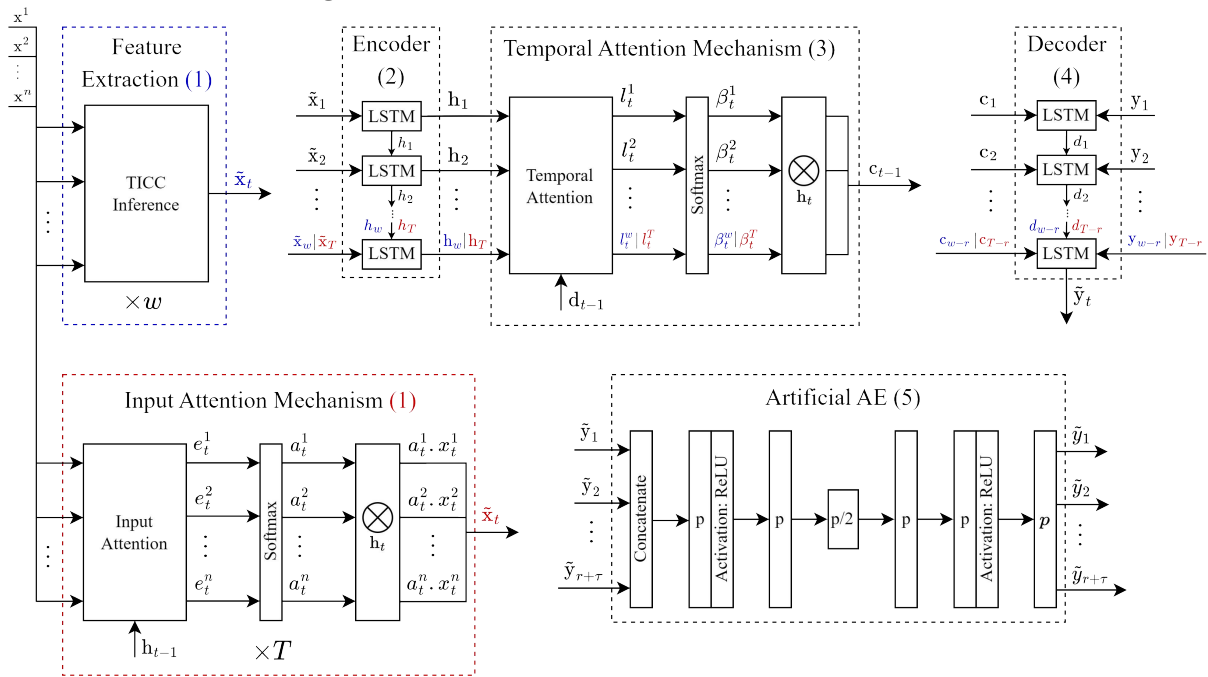
### 3.1.2 Experimental evaluation

The experimental evaluation consists of a comparison between 2 custom DA-RNN architectures, as state-of-the-art NARX models for processing multivariate time series, and 2 derived models, modified to receive the proposed feature spaces as inputs, for target time series reconstruction, with a range of  $r$  values, and forecasting, with a range of  $\tau$  values. The comparison is performed between models with similar numbers of parameters, intending to approximate the processing time for the analysis of an accuracy ramp, obtained by deepening each architecture.

#### 3.1.2.1 Models and parameter settings

The custom DA-RNN model contains the original DA-RNN structure with a customized definition of encoder hidden states  $m$  and decoder hidden states  $p$ . The derived model replaces the input attention with the proposed TICC feature extraction, which outputs a set of clustered environment states with equal relevance to the encoder at time  $t$ . Additionally, an artificial AE was tested and applied as a final structure in improving target value accuracy, receiving the latent outputs  $(\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_{r+\tau})$ . Figure 7 shows the custom and derived architectures. The main processes are presented in dashed boxes identified by sequential numbers; additionally, red and blue colors present inputs, processes and outputs exclusive to custom and derived models, respectively. The evaluation parameters are shown in Table 1.

Figure 7 – Custom and derived DA-RNN architectures.



Source: Own Authorship (2022).

Table 1 – Feature extraction evaluation parameters.

Models	$T$	$w$	$m = p$	$r$	$\tau$
Custom 1	20	-	5	0	5
Custom 2			8		
Derived 1	-	5	9		
Derived 2			14		

Source: Own Authorship (2022).

### 3.2 Model Predictive Controller

The proposed MPC is a method and algorithm designed for the regulation of the coolant temperature over a base controller performance and thermal response of the target. Vehicle performance, fuel efficiency and emission improvements can be achieved with the application of machine learning and time series analysis techniques. Forecasting strategies are used to predict coolant temperature and thermal impact states for a future control horizon, defined to allow a reinforcement learning labeling model to perform searches for optimal fan speeds. Fan speed demands are generated based on a thermal response system that predicts coolant temperature states with confidence based on cross-time correlations with thermal impact variables, resulting in less deviance from configured temperature set points, compared to classic controllers. Additionally, a model-based clustering strategy is used to generate unique thermal impact scenarios that, as

low-dimensional, yet intuitive, feature spaces, complement the environment states returned to the agent, improving its knowledge over thermal impacts in the target, and are fed to an NN controller, simplifying its architecture for coolant temperature regulation from general and simple thermal impact information. The selection of model-based clustering parameters can be leveraged for distinct vehicle configurations that present similar cross-time correlations between coolant temperature and thermal impact variables.

### 3.2.1 Coolant temperature regulation architecture

The proposed MPC method, strategy and algorithms are presented with two main structures, an internal structure, processed by the fan control processing unit, and an external structure to the vehicle, used to configure and train the controller for the real-time temperature regulation. The internal structure contains two main functions, forecasting and controlling. The data-driven structure and model design are shaped by the selection of optimal thermal impact variables on the coolant, from a range of available in-vehicle sensor data. Different data assignments are made based on observations over the variance of individual thermal impact series on short time sequences for real-time control purposes. As a data-driven method, similar data and machine learning model considerations are applicable to vehicle configurations that present the same set of real-time sensor data.

The external structure's design reflects the need for optimal fan speed demands that lead to cooling performance improvements from base observations. For this achievement, an RL labeling strategy, inspired by the online optimization problem of general MPC in setting the regulation objective and general interactive strategy, is defined. The internal structure is designed for the real-time regulation of the coolant temperature, inspired by the offline interactive strategy adopted in the RL labeling method. Initially a forecasting strategy is used to propagate all relevant thermal impact information to a future horizon, upon which, an NN control model predicts optimal fan speed demands for regulation. For this task, a predicted or fixed array of fan speed demands may be supplied to the controller in a subsequent iteration, according to the complexity of the fan hardware constraints and its appropriate learning by the controller NN model. Figure 8 presents the proposed MPC method, composed of forecasting and future horizon strategies for the offline training and inference of the controller. The controller training is performed after optimizing the RL model samples' inputs with cumulative actions in labels. An inference represents the acquisition of past  $t-d|t-n$  reference  $r$  and thermal impact  $x$  data,



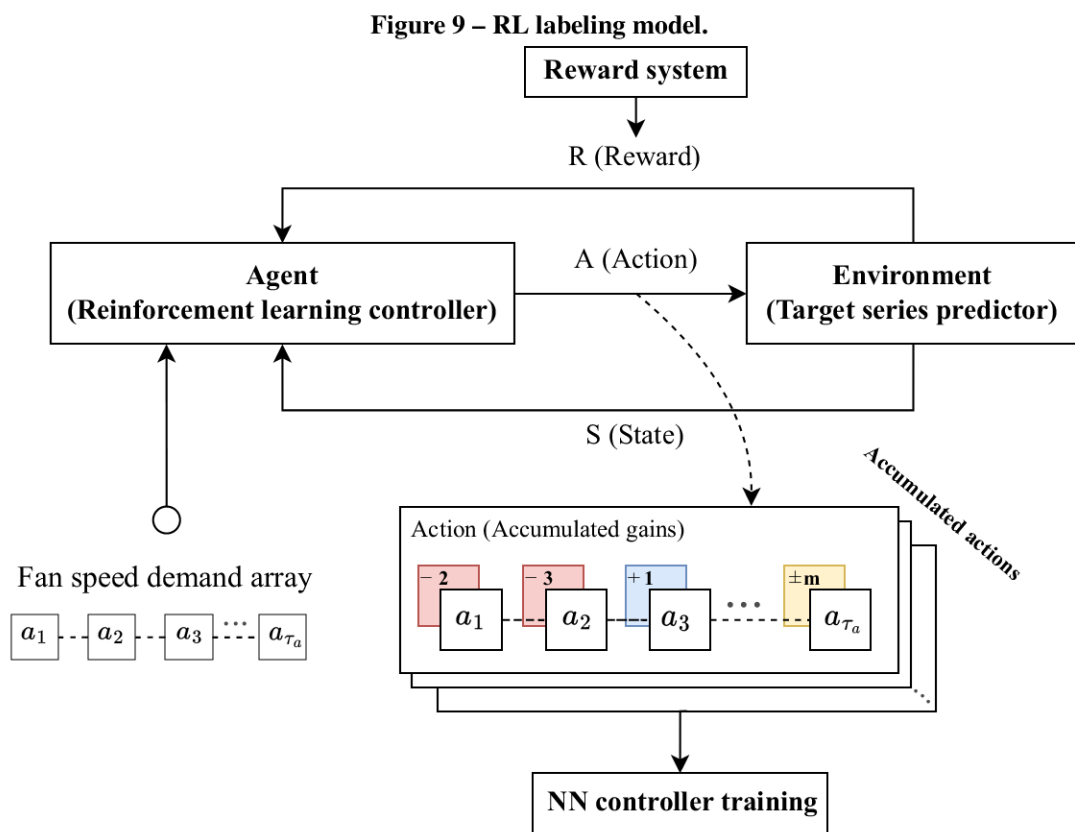
### 3.2.2 Reinforcement learning labeling

With the overall goal of searching for optimal fan speed demands, the definition of an RL model, designed over the coolant temperature regulation problem, enables the development of a prediction-based interactive scenario that, in advantage over the general MPC formulation, supports offline optimization. The offline optimizer (RL agent) solely relies on predicted states, without the need for system identification and linearization, given that the optimizer, now model-free, learns high-order patterns from observations that lead to the objective. Thus, the achievement of regulation objectives in highly nonlinear systems relates to the capability of learning by the RL agent while the environment, as a system response predictor, leverages techniques such as the downsampling of high-dimensional inputs with the goal of supplying discrete system responses to the iterative training of the agent. As an advantage of the offline optimization, the RL environment can leverage DNN features with a heavy focus on prediction accuracy. Linear constraints can be easily imposed with environment and state space rules as bounds for the selection of allowed actions and the labeling objective can be achieved as, concurrent with the iterative learning, a cumulative gain configuration  $U$  leads a predicted target sequence  $Y$  to a minimum accepted error from the target set point. This achievement, for a selection of relevant samples, is then concluded with the replacement of each original sample's sequential fan speeds with the new RL optimized values. Design efforts are then shifted to the internal structure controller in regards to training and inference capabilities.

#### 3.2.2.1 RL labeling interactive setting for coolant temperature regulation

For the purpose of labeling data to an NN controller, the RL model is set as episodic, covering a range of relevant samples for optimization. The samples are defined as multivariate time series  $X = \{s_1, \dots, s_W\}$  set with the same thermal impact variables, of size  $n$ , from the values adopted in evaluating the proposed feature extraction method, as the objective of predicting the coolant temperature is shared by the proposed RL environment. The sample's window size  $W$  was selected dynamically with the accuracy goal and architecture design of the RL environment, as a target prediction DNN. For this interactive setting, each episode is composed of a maximum number of iterations  $T$ , allowing the agent to iteratively learn from a given sample's static and dynamic environment states, which are composed of static thermal impact and dynamic fan speed and coolant temperature series. For the definition of an exploratory and exploitative agent,

actions are specified by the possibility of performing changes in the action space from a reference position  $P_t$  by imposing the addition or subtraction of a fixed gain  $k$ , for the simplicity of the action space and constrained environment, and, in each iteration, modifications are made in the current sample's inputs for the fan speed, according to the agent's choice of actions, which are then imposed on the environment, as a DNN predictor for concurrent and future target series values, simulating the coolant temperature response and providing the environment states that compose the state space returned to the agent. Figure 9 presents the proposed RL model architecture.



Source: Own Authorship (2022).

As shown in Figure 9, the offline controller (agent) performs actions in an action space  $A = \{a_1, \dots, a_{\tau_a}\}$  composed of an array of discrete values  $U = \{u_1, \dots, u_{\tau_a}\}$ , as sequential fan speed demands, and discrete positions  $P = \{p_1, \dots, p_{\tau_a}\}$ , where  $A_t = [U_t, P_t]$ , for  $t = \{1, 2, \dots, \tau_a\}$ , and  $\tau_a \leq W$ , for the possible dynamic association between position and gain for each action. In this way, the defined action space allows the agent to perform actions, from a reference position, fixed at the start of an RL episode or defined as a consequence of a previous interaction, that are weighted by the temporal relevance of action space positions in regards to the impact of applying a positive or negative gain  $k$ . The adopted action setting is shown in Table 2.



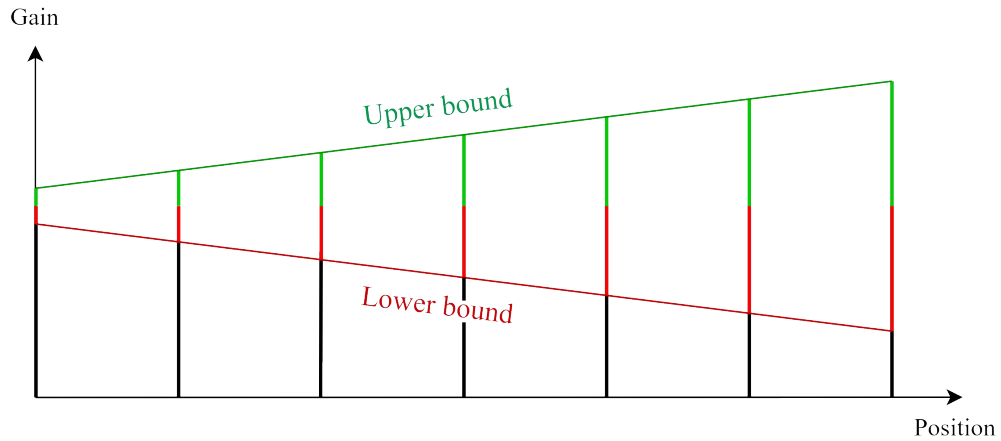
**Table 2 – Action setting.**

ID	Description	Summary
#1	Increase the fan speed	$+k$
#2	Decrease the fan speed	$-k$
#3	Move forward 2 positions	$+2p$
#4	Move back 2 positions	$-2p$
#5	Move forward 1 position and increase the fan speed	$+p + k$
#6	Move forward 1 position and decrease the fan speed	$+p - k$
#7	Move back 1 position and increase the fan speed	$-p + k$
#8	Move back 1 position and decrease the fan speed	$-p - k$

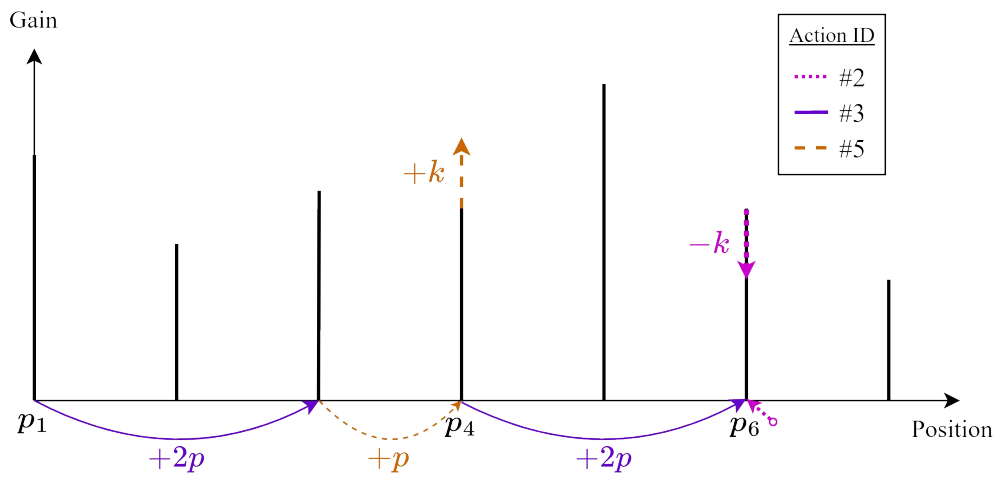
**Source: Own Authorship (2022).**

In Figure 10, visual representations of the defined action space are presented as a constrained environment (a) and an example strategy (b). As the agent makes observations over concurrent and future states of the target series, strategies may be adopted in weighting sequential interactions to increase or decrease the overall value of the action composition, leading to unfeasible or impossible actions that can be bounded in a constrained environment. Upon reaching these conditions, a programmatic elimination of actions can lead an agent's transfer of relevance to bounded actions, optimally generating improved and bounded strategies. Along with this representation, subsequent RL iterations are displayed in presenting one possible strategy adopted by the agent for tuning action values at two nonadjacent positions, an achievement that requires few RL iterations in the designed RL interactive loop. In example (b), 4 interactions, starting at  $p_1$ , are required to tune the gains of  $p_4$  and  $p_6$  with the use of movement-based and gain-based actions while taking advantage of the action space in learning the constrained action space environment. In addition, with respect to both representations, the agent's iterative strategy may assume an optimal path toward the constrained bounds, as in (a), tending toward a fine-tuning operation, as seen in (b), for exploring and exploiting fan speed configurations for regulation.

**Figure 10 – Visual representations of the action space.**



**(a) Constrained environment**



**(b) Strategy example**

Source: Own Authorship (2022).

In leveraging the learning of predicted impacts at every possible reference position  $P_t$  in the action space, the environment's DNN is designed to predict a target value horizon  $\tau_e$  that, for  $\tau_e = 2\tau_a - 1$ , given  $\tau_a > 1$ , enables a reward function  $r$ , shaped by the regulation objective, to assume a fixed horizon of future observations  $\tau_r = \tau_a - 1$  from a sliding window strategy, as follows:

$$r(u_i, p_i) = f([u_{i+1}, p_{i+1}], \dots, [u_{i+\tau_r}, p_{i+\tau_r}]) \quad (25)$$

Here, for  $i \leq \tau_a$ , the reward shaping can make use of the transition from a concurrent horizon of target observations, from the action space at  $p_1$ , to a future horizon, at  $p_{\tau_a}$ . A dynamic reward horizon  $\tau_r$  can also be applied with the following reward shape:

$$r(u_i, p_i) = f([u_{i+1}, p_{i+1}], \dots, [u_{\tau_e}, p_{\tau_e}]) \quad (26)$$

where, for all positions, the complete horizon defines the iterative objective. For the defined optimization problem, the reward horizon can be optimally shaped by the environment's static states, which dictate whether manufactured dynamic states, imposing a minimum or maximum possible actuation of the fan agent, constrained by the fan hardware, are able to inflict the change from a higher temperature, compared to a defined set point, to a lower temperature, or the inverse, at a given critical action space position. If possibly obtained for a given sample, this critical position can be dynamically defined, at each episode, as a starting position  $P_s$ , from which the regulation objective can be established. This reward strategy, adopted for the coolant temperature regulation problem, is shaped as follows:

$$r(u_i, p_i) = \begin{cases} f([u_i, p_i], [u_{i+1}, p_{i+1}], \dots, [u_{\tau_e}, p_{\tau_e}]) & i = P_s \\ 0 & \nexists P_s \end{cases} \quad (27)$$

where if  $P_s$  is nonexistent, the sample is discarded as the manufactured  $e_d$  solution is optimal.

Considering that the labeling objective is not constrained by sample or iteration characteristics, as a regulation goal must always be met despite the thermal impact setting of a given sample and iteration, the state space, for updating the agent with information over time series correlations in the target series, is set to cover the dynamic and static environment's states. While dynamic states are composed of the sum of changes observed in an interaction, in the form of cumulative actions  $U = \{u_1, \dots, u_{\tau_a}\}$  and predicted target series  $Y = \{y_1, \dots, y_{\tau_a}\}$  for a concurrent and future horizon, static states dictate the relevance of each episodic change for the overall labeling goal by comprising information of the thermal impacts in the target, exclusive to each sample and containing the previous fan speed, coolant temperature and thermal impact variables. This is achieved with the application of the proposed human-interpretable feature extraction process in generating a clustered environment representation of  $K$ -cluster likelihoods  $LL = \{LL_1, \dots, LL_K\}$ , as simple and robust states, for understanding each sample's relevance in the agent's decision over actions. Additionally, the current action position  $P_t$ , in reinforcing the agent's capability of weighting movement-related actions toward the regulation objective, and the starting position  $P_s$ , in defining the episodic reward horizon, are attached to the state space in enabling, in association with the defined action setting, the learning from a complete exploration of the environment and later exploitation of appropriate strategies, and the state space

observation for an adequate reward distribution, respectively. For a discrete time step composition  $t = \{1, 2, \dots, T\}$ , where  $T$  is the number of episodic iterations, the state space assumes the form  $S_t = [U_t, Y_t, LL, P_t, P_s]$ , where  $S_t \in S$  and  $S$  is the collection of all state spaces.

The reward function, shaped by the objective of achieving an optimal regulation in the general labeling setting, is set by a regression problem, where the interactive cost-minimization success is attributed to the agent's ability to reduce the deviation of the residuals from the discrete target  $Y$ , as the predicted coolant temperature, in reference to the equivalent horizon of set points  $\hat{Y}$ , as the Mean Squared Error (MSE). In assuming a reward distribution strategy, the adopted cumulative gain composition denotes that optimal strategies, leading to new and improved action compositions, could also have been built upon achievements from previous iterations. In this sense, strategies that serve as foundation for its subsequent have equal relevance toward the objective and, in addition, the learning of such strategy may be related to few actions in association with a high number of iterations, for a great but simple evolution in the action composition, or related to more specialized strategies, requiring a broader set of actions and few iterations but evolving into a complex composition. For this problem statement, a cumulative reward function, for a distribution of rewards that encourages a balanced exploration of complex strategies, is adopted as follows:

$$R_t = \sum_{t'=t}^T r_{t'}, \quad (28)$$

where the episode duration ranges from time  $t$  to  $T$ .

For labeling all available samples, reward states receive an interactive update, proportional to the cost-minimization error in the current time step  $t$ , independent of previous states, which enables a flexible control over sample changes in training. Additionally, it enables multiple labeling attempts of samples used in early training, where an increase in the sum of rewards can be observed as the agent optimally shifts to an exploitative strategy. Thus, with a distribution of rewards determined at every iteration from its related state space, for each positive reward milestone, as a partial valuable reward, the action space composition  $A_t$  can be saved for a future episode in the same sample. Algorithm 2 presents the proposed interactive setting with the described reward distribution for optimizing fan actuation and extracting labels.

---

**Algorithm 2 – Optimize fan actuation and extract labels**


---

**Given**  $X$  = collection of samples,  $U_i$  = base action composition,  $P_i$  = reference position,  $\tau_a$  = length of the action space,  $\tau_e$  = length of the target horizon,  $indexU$  = index of the agent series,  $indexY$  = index of the target series,  $W$  = sample window size,  $goalError$  = goal MSE,  $r_{max}$  = maximum reward,  $s$  = number of valuable rewards.

**Initialize**  $reward = 0$ ;  $d = \frac{r_{max}}{s}$ ;  $r\_list = [r(0), \dots, r(s)]$ , for  $r(n) = d * n$ ;  $\hat{Y}$  = list of  $\tau_e$  target set point values.

```

1  for each  $X_i \in X$  do
2     $Y_i = DNNEnvironmentInference(X_i, U_i)$ 
3     $LL = GetClusteredEnvironment(X_i)$ 
4     $U_{last} = X_i[indexU][W - \tau_a - 1]$ 
5     $Y_{last} = X_i[indexY][W - \tau_a - 1]$ 
6     $U_c = GetConstrainedActionComposition(U_{last}, Y_{last}, \hat{Y})$ 
7     $Y_c = DNNEnvironmentInference(X_i, U_c)$ 
8     $P_s = GetStartingPosition(Y_c)$ 
9     $S_t = [U_i, Y_i, LL, P_i, P_s]$ 
10    $milestone = oldMilestone = 0$ 
11    $baseError = MSE(Y_i[P_s, \dots, \tau_e], \hat{Y}[P_s, \dots, \tau_e])$ 
12   for  $t = 1, \dots, T$  do
13      $U_t, P_t = DQNAgentInference(S_t, reward)$ 
14      $Y_t = DNNEnvironmentInference(X_i, U_t)$ 
15      $S_t = [U_t, Y_t, LL, P_t, P_s]$ 
16      $currentError = MSE(Y_t[P_s, \dots, \tau_e], \hat{Y}[P_s, \dots, \tau_e])$ 
17      $proximityScore = \frac{(currentError - (2 * baseError)) * (2 * r_{max})}{(goalError - (2 * baseError))} - r_{max}$ 
18      $reward = \frac{milestone}{Abs(milestone)} * r\_list[Abs(milestone)]$ 
19     while  $-s < milestone < s$  do
20       if  $milestone \geq 0$  then
21         if  $proximityScore \geq (r\_list[milestone] + d)$  then
22            $reward += r\_list[1]$ 
23            $milestone ++$ 
24         else if  $proximityScore \leq (r\_list[milestone] - d)$  then
25            $reward -= r\_list[1]$ 
26            $milestone --$ 
27         else
28           break
29         end if
30       else
31         if  $proximityScore \geq (-r\_list[-milestone] + d)$  then
32            $reward += r\_list[1]$ 
33            $milestone ++$ 
34         else if  $proximityScore \leq (-r\_list[-milestone] - d)$  then
35            $reward -= r\_list[1]$ 
36            $milestone --$ 
37         else
38           break
39         end if
40       end if
41     end while
42     if  $milestone > oldMilestone$  then
43        $oldMilestone = milestone$ 
44        $L_i = GetLabel(X_i, U_t)$ 
45     end if
46   end for
47   return  $L_i$ 
48 end for

```

---

Source: Own Authorship (2022).

In addressing the discrete action and state spaces, Q-learning is applied. Thus, the action value function is defined as follows:

$$Q(s, a) = E [R_t | s_t = s, a_t = a, \pi], \quad (29)$$

where, through iterating, the optimal action value function is obtained. Additionally, due to the dimension of the state space, the DNN's feature extraction capabilities can be leveraged with the use of a DQN agent in estimating the optimal Q-function, as in (23).

### 3.2.3 In-vehicle model framework and strategy

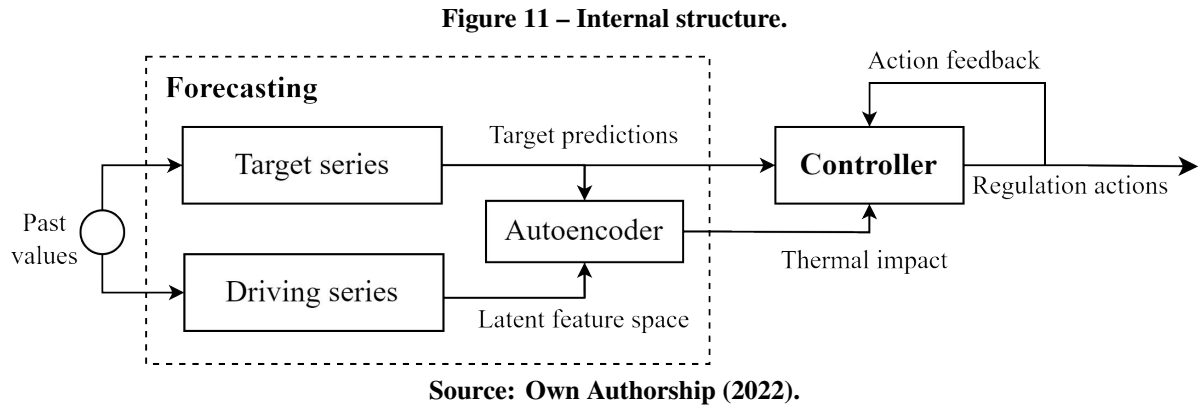
The internal structure, inspired by the RL labeling setting, was designed to reflect the gains from optimal actuation labels in real-time while addressing accuracy and processing time concerns for a real-time operation. To achieve this objective, the product of the correlation of sequential and multivariate target impact information is used to predict the target's future behavior as well as to enable the appropriate understanding of a more complete thermal setting for regulation. For this purpose, the proposed in-vehicle model framework is presented with two integrated strategies that aim at providing and using the optimal knowledge of the plant for the regulation of the refrigerant fluid.

A forecasting strategy is set with the provision task, as inspired by the RL environment's provision of target values to the agent, featuring the prediction of the target series, as the coolant temperature, driven by internal and external exogenous variables, and its related thermal feature space, to supply the optimal knowledge of the plant, with an adequate dimension for time efficiency, for the regulation action predictor, as the in-vehicle controller model.

As the RL model labels data for samples linked to episodes, TICC feature spaces  $LL$  are set to cover a known range of multivariate and sequential values that remain unchanged for an episode duration and, for such a process, the proposed feature extraction algorithm can be used once every RL episode in describing this static sample-defining state for the agent. Distinctly, the in-vehicle controller, as an NN model for the translation of relevant states into regulation actions, is set to predict a range of optimized controller actions with a single pass through the network, performed on inputs that contain unique clustered environment spaces at each iteration and with processing expenses relieved of past attention efforts that are leveraged by the forecasting strategy.

For each inference of the in-vehicle model, the forecasted outputs feed an NN controller

for the prediction of optimized fan speed demands and, similar to how action space observations ( $A_t \in S_t$ ) were required for the agent's knowledge over a constrained environment, a range of previously imposed fan speed demands  $x_u$  can be fed to the controller for a complete knowledge over imposed bounds. Figure 11 shows the internal structure in terms of forecasting and controller models. The parallel forecasting models' outputs are encoded as the horizon of target states is part of the feature space.



### 3.2.4 Experimental evaluation

The evaluation consists of the analysis of training results, presented according to each proposed ML model's objective, and an experimental and comparative physical evaluation of the complete controller strategy, making use of ML development environments, automotive calibration software and an Application Programming Interface (API) for integration. The following sections comprise the ML models' design and choice of parameters, highlighting the proximity between the RL labeling setting and in-vehicle architectures.

For initializing the dataset and model preparation as fit for the physical comparison, the sampling time was set for 0.1 seconds, where each adopted sequential length could be multiplied to acquire the temporal length or distance in each development stage.

#### 3.2.4.1 In-vehicle architecture and parameter settings

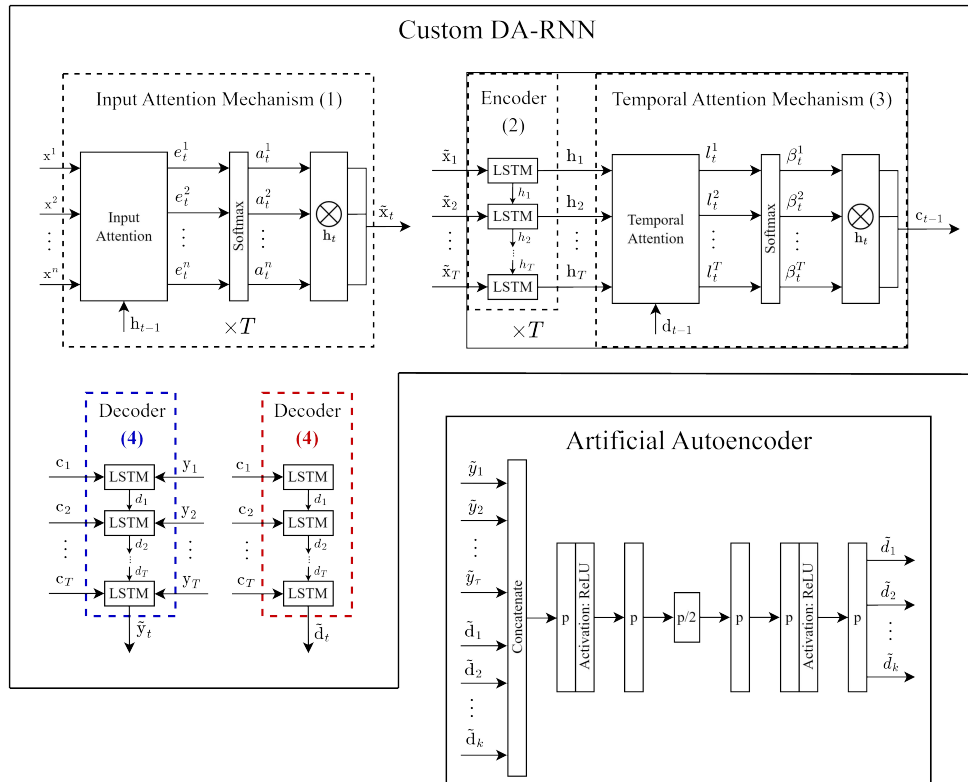
The RL labeling and in-vehicle models were designed with strategies for the selection of optimal parameters in weighting the accuracy and speed for each specific task. For accurately predicting target values that relate to exogenous time series, the architectures were built as inspired by the DA-RNN architecture, with decision over adopting specialized attention mechanisms for

precisely downsampling high-dimensional time series features into a horizon of target values. As applied in evaluating the proposed feature extraction method, strategy and algorithm, an artificial AE scaled with the size of the DA-RNN hidden states was tested at the end of each custom DA-RNN architecture to improve the prediction accuracy.

The in-vehicle forecasting models were designed with balance between accuracy and speed for the vehicle's real-time processing. The appropriate number of parameters for each model was defined over the size of the encoder  $m$  and decoder  $p$  hidden states, window size  $T$  and prediction horizon  $\tau$ . Given the connection between forecasting models, as seen in Figure 11, the AE parameters were set to scale with the size of the hidden states of the driving series forecasting model.

For simplicity, given that the DA-RNN temporal attention mechanism naturally captures long-term temporal dependencies of time series while selecting the most relevant input features (QIN *et al.*, 2017), the hidden states were selected with equal size in conducting a grid search over  $m = p \in \{4, 8, 16, 32, 64, 128\}$ , similar to (QIN *et al.*, 2017). Figure 12 presents the forecasting architecture of the target and driving series prediction models with decoders in blue and red, respectively. The values of  $m = p$  and  $T$  are presented in Table 3.

**Figure 12 – Forecasting architecture.**



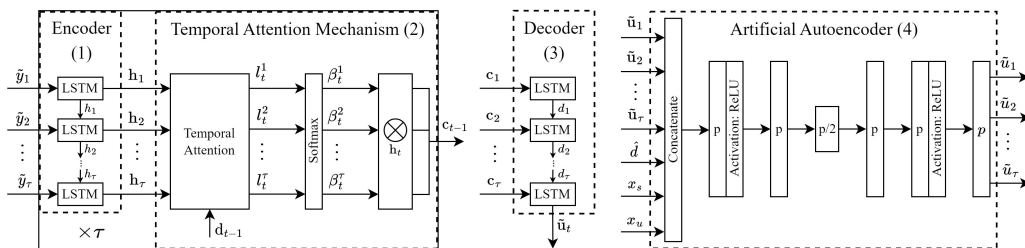


Considering the forecasting model's target  $\hat{y}_f$  as the controller's model reference  $r_c$ , the controller was designed with relieved attention expenses as the fan actuation is predicted through downsampling a single target series with future temporal relevance. The input attention mechanism was removed, as there is no need to compute the multivariate relevance of the input series at each time step, and the temporal attention mechanism was reduced, as past regulation target values, often available to condition the decoder, were used to forecast the driving series feature space of thermal impacts. With such a reduced model complexity, the controller assumes the task of translating the forecasted coolant temperature series  $r_c$  into the optimized fan actuation series  $\hat{y}_c$  and, given the use of extracted thermal impacts, the decoded outputs are concatenated with relevant thermal impact variables.

The dynamic set of thermal impact variables  $X_d$  was selected as equal to those defined in the proposed human-interpretable feature space evaluation, where, TICC parameters, weights, vehicle configuration, driving session and engine load conditions could be shared in training and testing the complete controller solution. In contrast, however, static variables  $X_s$  were defined and applied in the controller due to their individual contributions in the thermal impact setting. This definition was disregarded for the feature extraction evaluation because cross-time correlations of observations with low variance, for an adopted TICC window size  $w$ , are not optimally captured.

For the training and physical evaluations of the control solution, the length of the forecasted coolant temperature and fan actuation horizons  $\tau$  was selected as equal to the length of the action space horizon  $\tau_a$ , making use of all labeled fan speed optimizations in the training of the controller model. The size of the hidden states was selected with a grid search over  $m = p \in \{4, 8, 16, 32, 64, 128\}$ , similar to the strategy adopted for the forecasting models. Figure 13 presents the controller architecture with predicted target series  $\hat{y}$ , driving feature space  $\hat{d}$ , acquired static variables  $x_s$  and previously imposed fan speed demand  $x_u$  as inputs. The values of  $m = p$  and  $\tau$  are presented in Table 3 with the complete size of the controller solution displayed in the sum of all in-vehicle model parameters.

**Figure 13 – Controller architecture.**



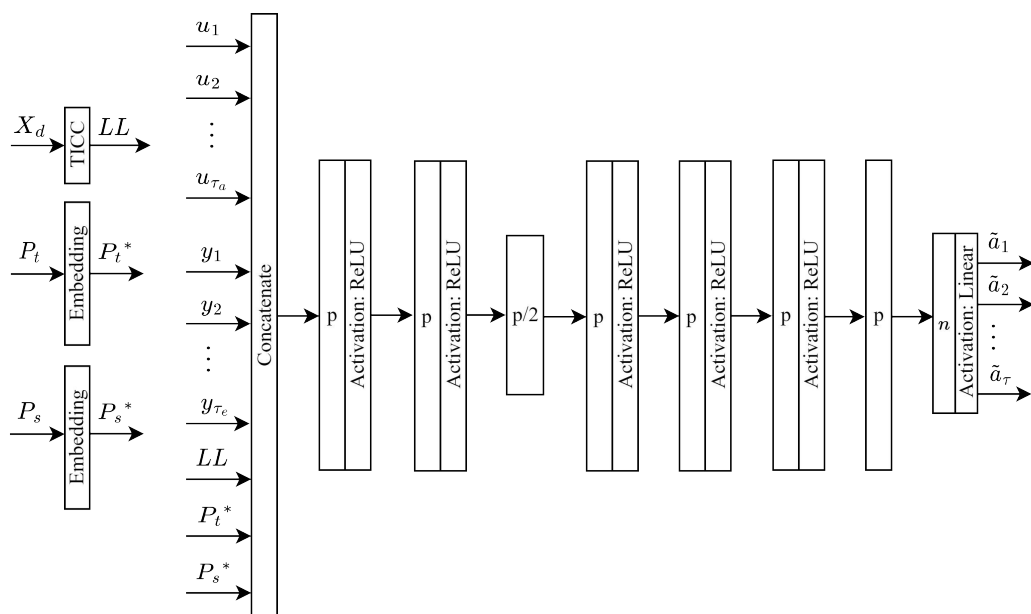
Source: Own Authorship (2022).

### 3.2.4.2 RL labeling architecture and parameter settings

While it shares the target prediction objective with the in-vehicle forecasting strategy, the offline RL labeling environment presents distinctions in regards to time efficiency in processing as it does not concern real-time operations and possible production costs. Thus, the environment's model architecture was designed over the custom DA-RNN for target forecasting, as shown in Figure 12, and, to improve the prediction accuracy at the cost of a higher dimension, a new grid search was conducted  $m = p \in \{8, 16, 32, 128, 256\}$ , starting with the adopted values for the in-vehicle target forecasting model with a fixed  $T$ , larger than the original value for an improved temporal context. The values were selected as the observed accuracy converged.

For the agent, which receives the complete prediction horizon of concurrent and future values  $Y_t$ , and a simple representation of past values  $LL$ , the architecture was designed as inspired by the adopted artificial AE for extracting the individual relevance of the target and action compositions, as weighted by the positional arguments and static context. The positional arguments were handled with an initial embedding layer, converting  $[P_t, P_s] \in \mathbb{N}$  to  $[P_t^*, P_s^*] \in \mathbb{R}$ , and a deeper decoder was tested and implemented due to improvements in the training convergence. Figure 14 displays the agent's model architecture, where,  $p$  is the value for scaling all the fully connected layers and  $\tau$  is the number of actions  $\hat{a}$ , as shown in Table 2. The rest of the parameters of all designed models are shown in Table 3.

**Figure 14 – Agent architecture.**



**Source: Own Authorship (2022).**

**Table 3 – MPC evaluation parameters.**

Models		$m = p$	$T$	$\tau$	parameters
Forecasting	Target series	8	26	10	8808
	Driving series	16		$k=4$	12788
Controller		8	10	$\tau_a=10$	6970
RL labeling	Environment	32	36	$\tau_e=19$	103935
	Agent	36	-	8	8378

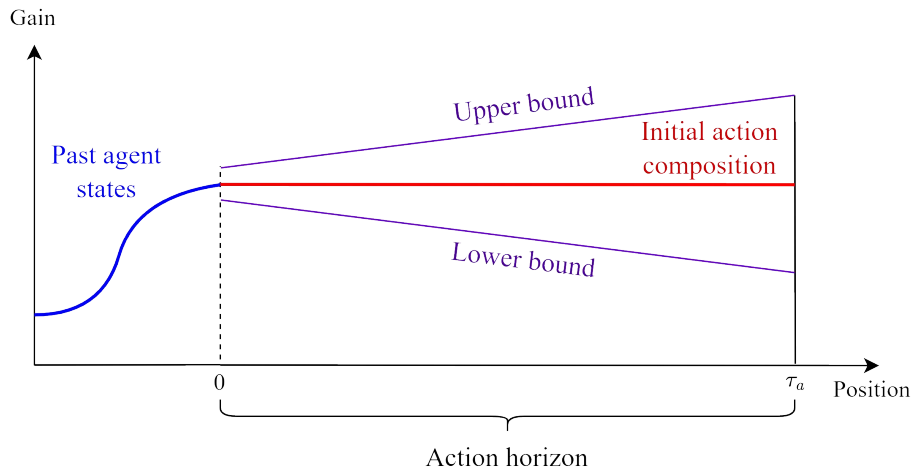
**Source: Own Authorship (2022).**

The action constraints were defined, in contribution with the OEM's thermal management team, as relating to a fixed limit  $l$  between the current  $u_t$  and immediately preceding  $u_{t-1}$  fan speed values. For a maximum or minimum actuation, the constrained composition assumes a linear form, as shown in example (a) of Figure 10, for both increasing or decreasing fan actuation. The constrained function for possible actions  $a_t$  is presented as follows:

$$a_t = [u_t, p_t] \quad \text{subject to} \quad |u_t - u_{t-1}| \leq l \quad (30)$$

where,  $l$  was defined as 50 rpm and the fixed gain  $k$ , applied in increasing or decreasing the fan speed value  $u_t = u_{t-1}$  at a given position  $p_t$  was selected as 5 rpm, representing a fraction of  $l$  in enabling the formation of complex action compositions inside the bounded area of actuation.

A strategy to provide a maximum area of actuation, for an optimal exploration of the environment, was selected according to the defined constrained environment for the initial action composition  $A_0$ , where all available action values  $U_0$  were set with the immediately preceding fan speed value  $u_{-1}$  as part of the static context returned to the agent. The initial action composition, previous to the first interaction and bounded by the adopted constrained function, is presented in Figure 15.

**Figure 15 – Initial action composition.**

**Source: Own Authorship (2022).**

### 3.2.5 Physical evaluation

After the definition, training and testing of the developed models, resulting in the collection of optimal weights, a comparative physical evaluation method, strategy and algorithm were developed for the complete analysis and study of possible gains with the proposed controller strategy. The physical evaluation was structured as experimental with strategies for the accurate access to similar partial engine load conditions on vehicle runs in a route, defined with similar conditions as the training and testing observations.

The performance metrics for comparison were defined with contributions from the thermal management team according to the level of interpretability and interest in the internal adoption of a new control strategy at a production scale. Thus, they were defined as fuel consumption and fan usage, with the general acquisition of available variables for the further development of experimental strategies.

Considering the need to visualize, acquire, process and impose data in real-time, the evaluation algorithm was developed as an integration of the ATI VISION software, used for the visualization, acquisition and calibration of the vehicle's Electronic Control Units (ECUs) parameters, and Python, used with resources from ML libraries for training, testing and predicting with all developed models. The algorithm was developed for a CAN Calibration Protocol (CCP) communication between an OEM's notebook, with adequate computational processing capabilities, and the vehicle's ECUs. Multiprocessing was used for the simultaneous execution of the acquisition of interest data, data processing, imposition of fan speed demands and data logging, while threading was used in the acquisition processes for the simultaneous collection of data. A time counter was used in each specific process and a time stamp of acquisition ( $t_a$ ) was shared for the synchronization of all dependant processes. A pool multiprocessing object was used to contain the processes while a pipe object was used to enable intercommunication between processes as needed.

The pool process for the acquisition of interest data contains a pipe object for sharing the acquired values with the controller and data processing pool. Initially, a flag is sent, from the data processing pool, to start the acquisition process, which is responsible for acquiring, arranging and sending data frames, specified to the requirements of all forecasting and controller models, to the controller and data processing pool. The arrangement of each data frame is made with a continuous collection of simultaneous values, synchronized for the defined sampling time  $t_s$  of 0.1 seconds. When a composition of  $T$  sequential values is made, according to the forecasting model

requirements seen in Table 3, the sample and  $t_a$  are sent to the controller and data processing pool for a complete algorithm synchronization. Algorithm 3 shows the acquisition process.

---

**Algorithm 3 – Acquire the data of interest**

---

**Given**  $t_s$  = sampling time,  $T$  = sequential length,  $i_{list}$  = list of data item names for acquisition in the vehicle's calibration file.

**Initialize**  $t = 0$ ,  $S_{list}$  = empty list.

```

1 while 1 do
2    $currentTime = GetTime()$ 
3   if  $t == 0$  or  $Abs(currentTime - t) \leq \frac{t_s}{10}$  then
4      $dataCollection = Thread(GetData(i_{list}))$ 
5     if  $t == 0$  then
6        $t = GetTime()$ 
7        $t = t + t_s$ 
8     else
9        $t = t + t_s$ 
10    end if
11    if  $Length(dataCollection) < T$  then
12      append  $dataCollection$  to  $S_{list}$ 
13    else if  $Length(dataCollection) == T$  then
14      return  $S_{list}, t - t_s$ 
15    else
16       $S_{list} = RemoveFirstElement(S_{list})$ 
17      append  $dataCollection$  to  $S_{list}$ 
18       $t_a = t - t_s$ 
19      return  $S_{list}, t_a$ 
20    end if
21  end if
22 end while

```

---

Source: Own Authorship (2022).

After the reception of the first data frame, the controller and data processing pool execute a complete pass through the forecasting and controller networks, generating fan speed demands for an imposition in the fan hardware. Algorithm 4 presents the controller execution.

---

**Algorithm 4 – Execute the controller inference**

---

**Given**  $S_{list}$  = sample from the data acquisition process,  $t_s$  = sampling time.

```

1 while 1 do
2    $S_{list}, t_a = ReceiveSample()$ 
3    $Y_{list} = TargetSeriesForecasting(S_{list})$ 
4    $LL_{list} = DrivingSeriesForecasting(S_{list})$ 
5    $U_{list} = ControllerInference(Y_{list}, LL_{list})$ 
6   return  $U_{list}, t_a$ 
7 end while

```

---

Source: Own Authorship (2022).

The values are then sent, along with  $t_a$ , to the imposition process pool, which, respecting the time synchronization from the currently acquired data frame, ensures that each output is correctly imposed in the fan hardware with modifications in the vehicle's calibration file. Algorithm 5 displays the imposition of the generated demands.

---

**Algorithm 5 – Impose the fan speed values**


---

**Given**  $t_s$  = sampling time,  $n_s$  = number of consecutive fan speed impositions from one sample.

```

1 while 1 do
2    $U_{list}, t_a = ReceiveDemand()$ 
3    $currentTime = GetTime()$ 
4    $t = t_a + t_s$ 
5    $idx = 0$ 
6   while  $idx < n_s$  do
7     if  $Abs(t - currentTime) \leq \frac{t_s}{10}$  then
8       return  $U_{list}[idx]$ 
9        $t = t + t_s$ 
10       $idx = idx + 1$ 
11    end if
12  end while
13 end while

```

---

**Source: Own Authorship (2022).**

While performing the acquisition, processing and imposition of the values of interest, a data logging process pool was used for composing and saving the physical evaluation dataset. To avoid overloading the communication system of each process, the synchronization, acquisition, data treatment and saving function were set as independent of the other processes. The separate data logging process is shown in Algorithm 6.

---

**Algorithm 6 – Log the data of interest**


---

**Given**  $t_s$  = sampling time,  $T$  = sequential length,  $i_{list}$  = list of data item names for logging in the vehicle's calibration file.

**Initialize**  $t = 0$ .

```

1 while 1 do
2    $currentTime = GetTime()$ 
3   if  $t == 0$  or  $Abs(currentTime - t) \leq \frac{t_s}{10}$  then
4      $dataCollection = Thread(GetData(i_{list}))$ 
5     if  $t == 0$  then
6        $t = GetTime()$ 
7        $t = t + t_s$ 
8     else
9        $t = t + t_s$ 
10    end if
11    return  $dataCollection$ 
12  end if
13 end while

```

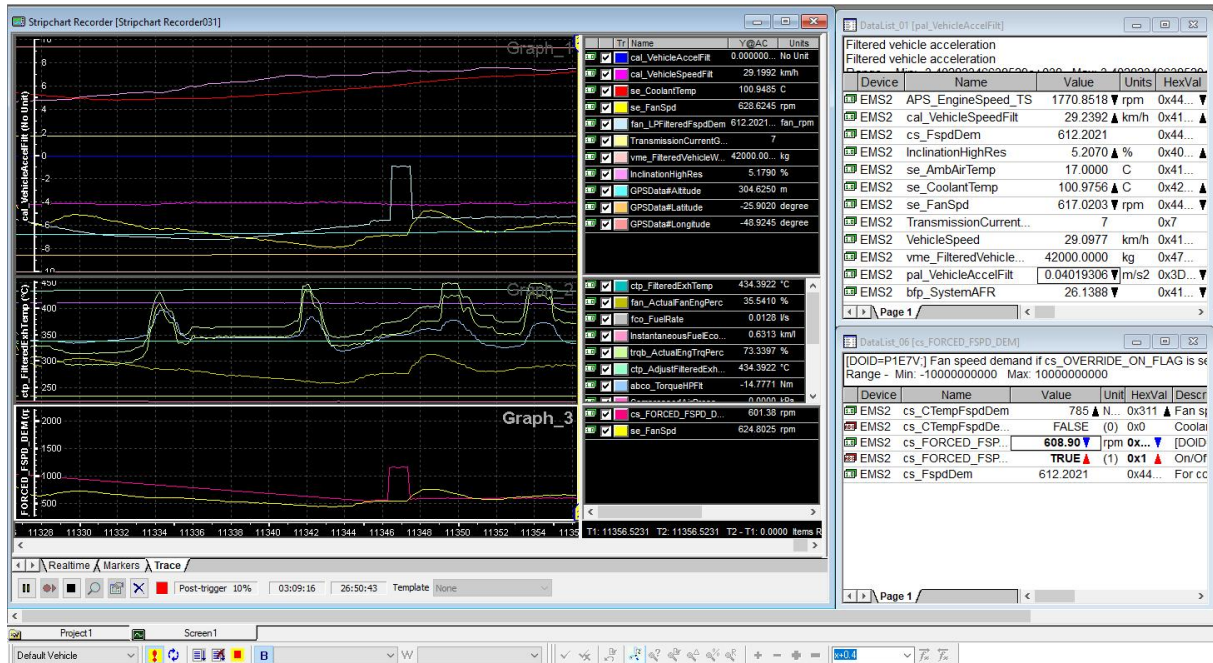
---

**Source: Own Authorship (2022).**

For the real-time visualization of each variable and to validate the imposition of fan speed demands in the calibration file, the built-in visualization tool of the ATI VISION software was used, with a custom configuration for the data visualization, including graphic representations and item distributions, dynamically defined to validate optimal conditions for a comparative analysis between control strategies. A custom ATI VISION visualization window, used for

the verification of interest variables and validation of the developed evaluation algorithm, is presented in Figure 16.

**Figure 16 – Configured ATI VISION screen for visualization.**



Source: Own Authorship (2022).

After the complete validation of the developed algorithm, the comparative analysis was structured, with support from the thermal management team, for the definition of the route, driving strategy and configuration for multiple runs with the adopted vehicle. The route was defined based on similarities with the one used for training and testing the complete control strategy, where, an uphill road configuration was adopted due to a longer partial engine load operation. The driving strategy was defined according to the route, where the driver was guided into maintaining the vehicle's speed constant at 35 km/h while avoiding gear changes.

## 4 RESULTS AND DISCUSSION

### 4.1 Feature extraction

This section describes the use of the proposed human-interpretable feature extraction process in an NARX problem, comprising the evaluation and study of the prediction accuracy of similar ML models that differ in the reception of each type of input. The problem was additionally defined as befitting with the purpose of evaluating the complete MPC controller strategy, where, TICC parameters, weights, vehicle configuration, driving session, engine load conditions and the summary of results were applied in training and testing the controller solution.

#### 4.1.1 Dataset preparation and feature extraction setup

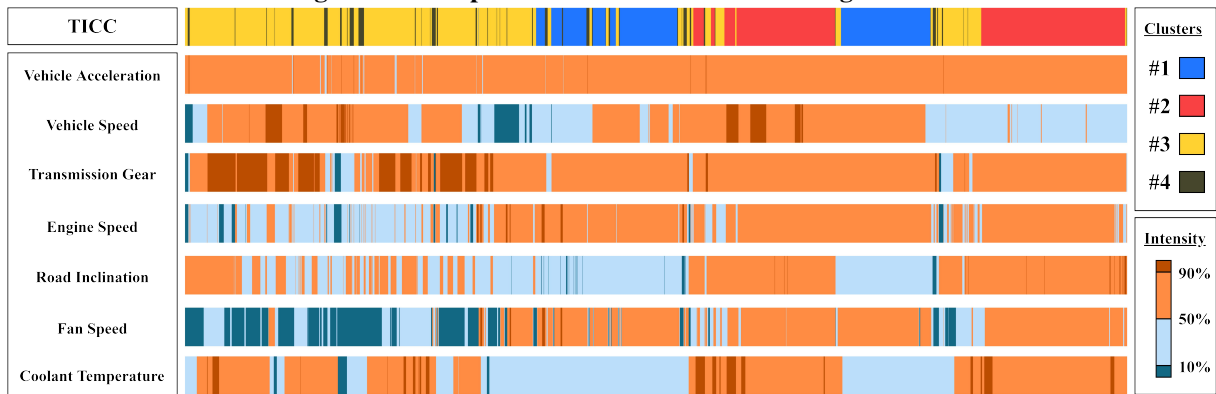
The evaluation was performed in a dataset, provided by an OEM, containing thermal impact-related sensor data, for distinct engine load conditions, from a real driving session. This session lasts approximately 2 hours and occurs on a road in a Brazilian city. Observations are made with 7 sensory measurements every 0.1 seconds:

- Vehicle Acceleration
- Vehicle Speed
- Transmission Gear
- Engine Speed
- Road Inclination
- Fan Speed
- Coolant Temperature

TICC (HALLAC; VARE, *et al.*, 2018) parameters were dynamically defined over subsequences of the dataset with the initial goal of visualizing reasonable homogeneity and consistency, in iterating for convergence, for the quantity of cluster samples. A custom visualization tool was built in Python in integration with HTML to support a dynamic definition of parameters over values and visual resources. A discrete sampling time of 0.3 seconds was adopted for a window size  $w_{0.3} = 6$  and number of clusters  $k = 4$ . Figure 17 shows the final visual outputs considered for the feature extraction setup. The first bar contains colors for each sample's cluster assignment, while the bar for each sensor highlights its related intensity at each timestamp.



**Figure 17 – Outputs of the TICC visualization algorithm.**



Source: Own Authorship (2022).

For the initial objective, visual outputs of the number of cluster samples at each iteration of the TICC optimization algorithm were used. Considering that an MRF is assigned to each cluster, all partial correlations between variables (sensors), as graphic edges, denote the temporal strength in the relationship between all defined sensors. Thus, for interpretation, the Betweenness Centrality (BC) scores (BRANDES, 2001), which summarize the contributions of each sensor in a clustered environment definition, were displayed in an "importance" matrix  $k \times n$ , where each value represents the impact weight of each sensor. In searching for an optimal clustered environment, homogeneity and consistency in assigning clusters were initially achieved by searching for a low Mean Average Deviation (MAD) among all sensor relationship weights for each cluster, as it contains the relative cluster importance at a given optimization iteration. Fine tuning is achieved by increasing the average BC score of the cluster with the least number of samples, inherently the least "important" cluster, while maintaining or improving the overall score proximity. The search for a higher average BC score, among all clusters, increases the confidence in each cluster assignment. Table 4 contains the BC score for each sensor within each defined cluster.

**Table 4 – Importance matrix.**

Clusters	Acc.	Spd.	Gear	Eng. Spd.	Road Incl.	Fan Spd.	Cool. Temp.
#1	84.033	0	0	255.55	41.516	137.506	39.396
#2	42.671	0	0	256.497	28.174	47.543	19.115
#3	50.503	67.539	39.996	179.106	53.944	65.774	43.138
#4	41.107	32.725	27.268	130.192	30.645	46.96	20.102

Source: Own Authorship (2022).

After the dynamic definition of initial parameters through the analysis of optimization BC scores, the visualization of all sample assignments optimally leads to a human-interpretable cluster definition and additional assignment smoothing. Figure 17 shows the outputs of a

visualization tool, presenting a range of each cluster's assigned samples and the intensity of each sensor for the specification of a human-interpretable clustered environment.

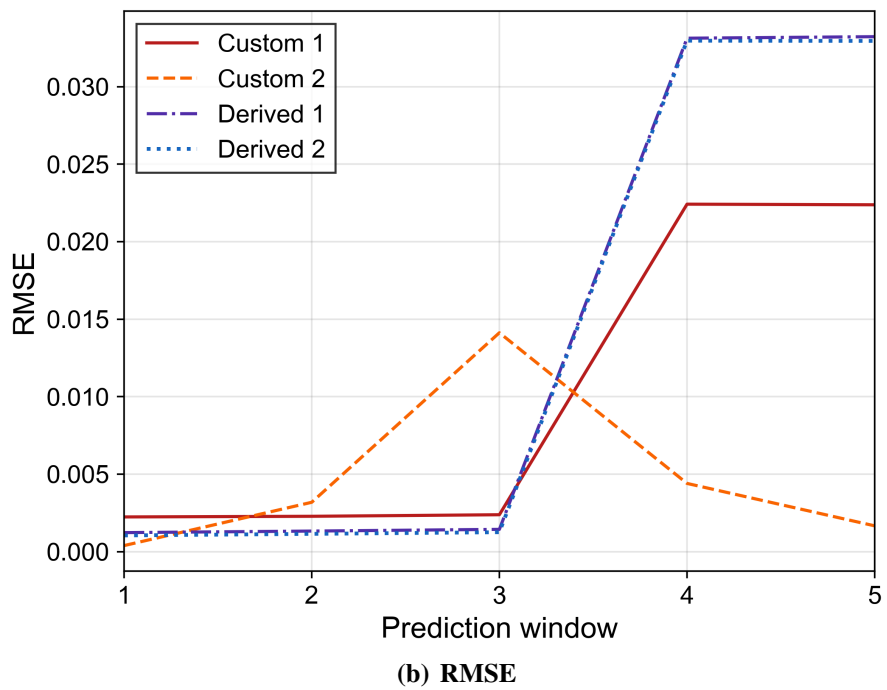
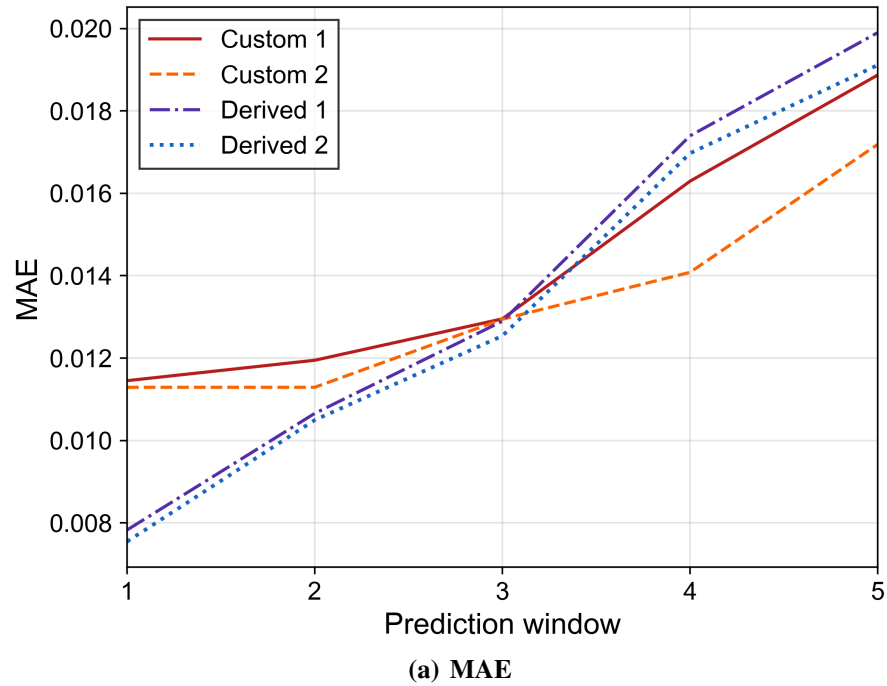
Table 4 and Figure 17 present the visual context for the final clustered environment definition. Table 4 shows that the engine speed sensor has a high correlation importance in the thermal performance of all clusters, while vehicle speed and transmission gear shifts tend to passively change, adapting to the environment. Cluster #1 carries the highest engine speed and fan speed scores, as fan speed demands are determined upon outer-loop requests, more independent of the coolant temperature response and with higher intensity than other clusters. Cluster #2 maintains a high engine speed importance with an average score among other sensors as the observed increase in heat rejection, from the vehicle and engine speed, road inclination and coolant temperature, correlates to a higher heat rejection demand. Clusters #3 and #4 present lower heat-rejection environments and a nonzero score for the vehicle speed and transmission gear shifts. The observed cluster attributes are a consequence of a more direct impact from driver control inputs in the vehicle's thermal response. Cluster #4 describes a more pronounced variation in sensor data in the occurrence of subsequent rapid gear shifts.

#### 4.1.2 Evaluation metrics and structure

To measure the effectiveness of the proposed feature extraction method, two different evaluation metrics are considered in comparing the prediction performance of models with a similar quantity of trainable parameters. For a number of test observations  $N$ , ground truth values  $y$  and predicted values  $\hat{y}$ , the Root Mean Squared Error (RMSE), defined as  $\mathbf{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i^i - \hat{y}_i^i)^2}$ , contributes to the comparative analysis with a higher weight in the impact of outliers, whereas the Mean Absolute Error (MAE), defined as  $\mathbf{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i^i - \hat{y}_i^i|$ , contributes with the application of higher weights to the overall performance in the test set without highlighting resulting outliers.

The adopted evaluation structure contains results for the overall performance of each model and a graphic presentation of results for each metric and predicted value, composing the prediction window. Thus, intending the discussion of a general and a more detailed view of each model's performance.

The comparative analysis is performed on the prediction error, with both metrics, of all 4 models. Figure 18 and Table 5 present the accuracy performance results.

Figure 18 – Error  $\times$  prediction window curve.

Source: Own Authorship (2022).

Table 5 – Evaluation results.

Metrics		Models			
		Custom 1	Derived 1	Custom 2	Derived 2
Number of Param.		3427	3169	7118	7106
Accuracy	MAE	0.0143	0.0137	0.0135	<b>0.0133</b>
	RMSE	0.0103	0.0141	<b>0.00475</b>	0.0139

Source: Own Authorship (2022).

Table 5 shows that the application of the proposed feature spaces, as a time series of equal and relevant clustered environment states, leads to forecasting NN models with a smaller absolute error. This improvement, in comparison to approaches that make use of specialized input attention mechanisms, demonstrates that the extraction of model agnostic features, available in a human-interpretable clustered environment, optimally supports the prediction of values from inputs, with a reduced dimension, that reflects the general dynamic of a given model. In the case of exceptions, as specific and rare responses to exogenous series, the interpretable and dynamic optimization process tends to suppress high-dimensional outliers, aggravating the RMSE metric performance. As outliers become apparent, the result of a greater number of training samples or better selection of high-dimensional spaces, the use of the proposed feature extraction method leads to additional improvements in the MAE metric, tending to outperform more specialized models in the RMSE metric.

For both architectures and choice of parameters, a deeper model results in performance improvements. Table 5 and Figure 18 show a significant decrease in RMSE in the case of the custom model, as the capability of describing high-dimensional outliers increases, whereas, for the derived model, a proportional decrease of MAE and RMSE is observed, befitting with a feature representation that carries interpretable correlation features, as the task of describing outliers, suppressed during optimization, remains challenging to deeper models. A more detailed view of predicted values, in Figure 18, shows that for both derived models, the proposed feature space better represents the impact of the driving series, as patterns, in forecasting values up to 0.3 seconds, consequence of more balanced, model agnostic and specific, features and model dynamics outliers, tending to decrease the performance as inadequately interpreted impacts in the target affect the accuracy performance of subsequent values.

#### 4.1.3 Evaluation summary

The feature space, as the likelihood that a sample belongs to the cluster in an interpretable environment, presents a reduced dimension with patterns that relieve expenses in the design of ML architectures. Furthermore, the application of dynamic programming eases the definition of parameters due to a direct visualization of its impacts while converging to an optimal solution. The experimental comparative evaluation results show that the proposed method of feature extraction relies on the optimal description of model dynamics, as the reduction of data outliers can improve the accuracy of ML prediction models, and, as the dynamics of a given system are

well described for a given objective, such as regulation, the feature extraction process becomes interesting for systems with limited processing capabilities, such as the internal structure of the proposed MPC control strategy, or to increase the time efficiency for optimization, as applied in complementing the RL agent’s state space with a simple feature representation. A promising future direction is to explore different human-interpretable methods for the dynamic clustering optimization process and the use of model-based clustering feature spaces for integrated systems and applications with limited processing capabilities.

## 4.2 Model predictive controller

This section describes the experimental evaluation of the proposed MPC controller solution, comprising the training results of RL labeling, forecasting and controller models, visually presented as fit for each objective, and the conception, results and study of an experimental and comparative physical evaluation between the proposed controller and a PID solution, applied at a production scale for the covered heavy-duty vehicle, in terms of coolant temperature regulation impacts on the vehicle performance.

### 4.2.1 Dataset preparation

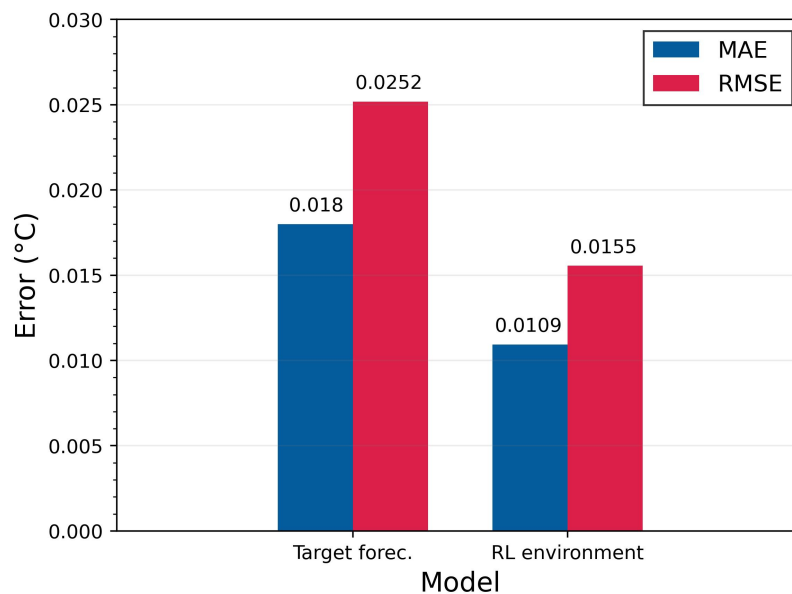
For training and testing the forecasting models, the dataset described in Section 4.1.1 was used with a split of 80% for training, 10% for validating and 10% for testing, where each sample was defined according to the input and output specifications of each model, as presented in Table 3. For the offline labeling setting, where the environment’s DNN model requires a separate training process for an unbiased performance over RL samples, the dataset and splits of the forecasting models were used in training and testing the environment’s DNN while a new dataset, with observations of an equal vehicle configuration and route, and similar ambient conditions, timespan and driving performance, was used to label the data for the controller.

Given that the controller’s data specification is a subset of the RL environment, as the input  $T$  and output  $\tau$  lengths are smaller, the RL labeling datasets were specified for the environment’s model due to simplicity and code optimization, later reframed to the controller parameters for its training.

#### 4.2.2 Training and testing

Due to the application of an ML strategy that employs attention mechanisms in weighting each driving series individually and temporally, the forecasting and environment models presented a stable conversion to a minimum loss. To more intuitively visualize the goals achieved in training, the weights, prior to the overfitting of each model, were saved and applied in measuring the prediction error, with the forecasting and environment models, and class accuracy, in the case of the driving series forecasting model. Figure 19 displays the error in the test set for the target series forecasting and environment models, where, the MAE and RMSE metrics were used for averaging the prediction accuracy and sensitivity to outliers.

**Figure 19 – Accuracy of the target series forecasting and environment models.**

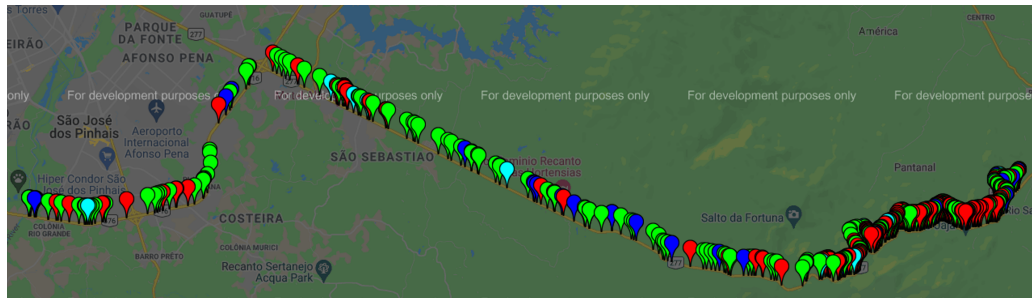


**Source: Own Authorship (2022).**

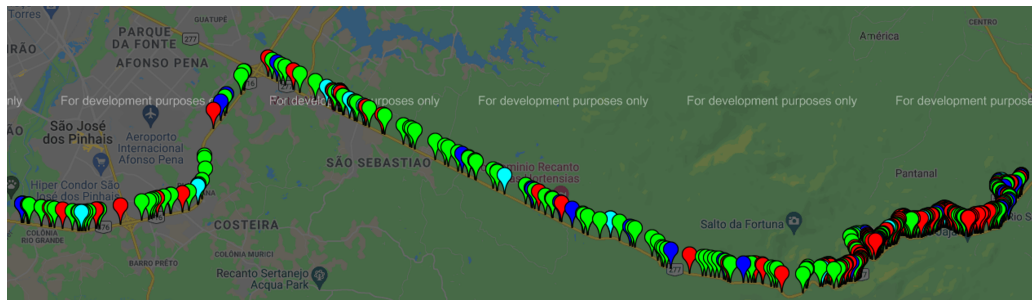
Due to the accurate response of each model, the optimized weights were deemed fit for each objective in providing accurate coolant temperature states to the controller and RL agent, where the use of a deeper architecture contributed to a greater accuracy and robustness to outliers.

To demonstrate the results of the driving series forecasting model after training, the predicted clustering feature spaces, composed of 4 cluster likelihoods, were displayed with colors for each most suitable cluster and arranged in Global Positioning System (GPS) coordinates, acquired for each test sample, for a simple visual comparison between the outputs of the TICC model and the driving series model predictions. Figure 20 presents the single-class accuracy of the model throughout the test set and Table 6 shows the multi-class and single-class accuracy in terms of MAE, RMSE, and Percentage of Correct Predictions (PCP).

**Figure 20 – Single-class performance of the driving series forecasting model.**



**(a) TICC outputs**



**(b) Prediction**

**Source: Own Authorship (2022).**

**Table 6 – Accuracy of the driving series forecasting model.**

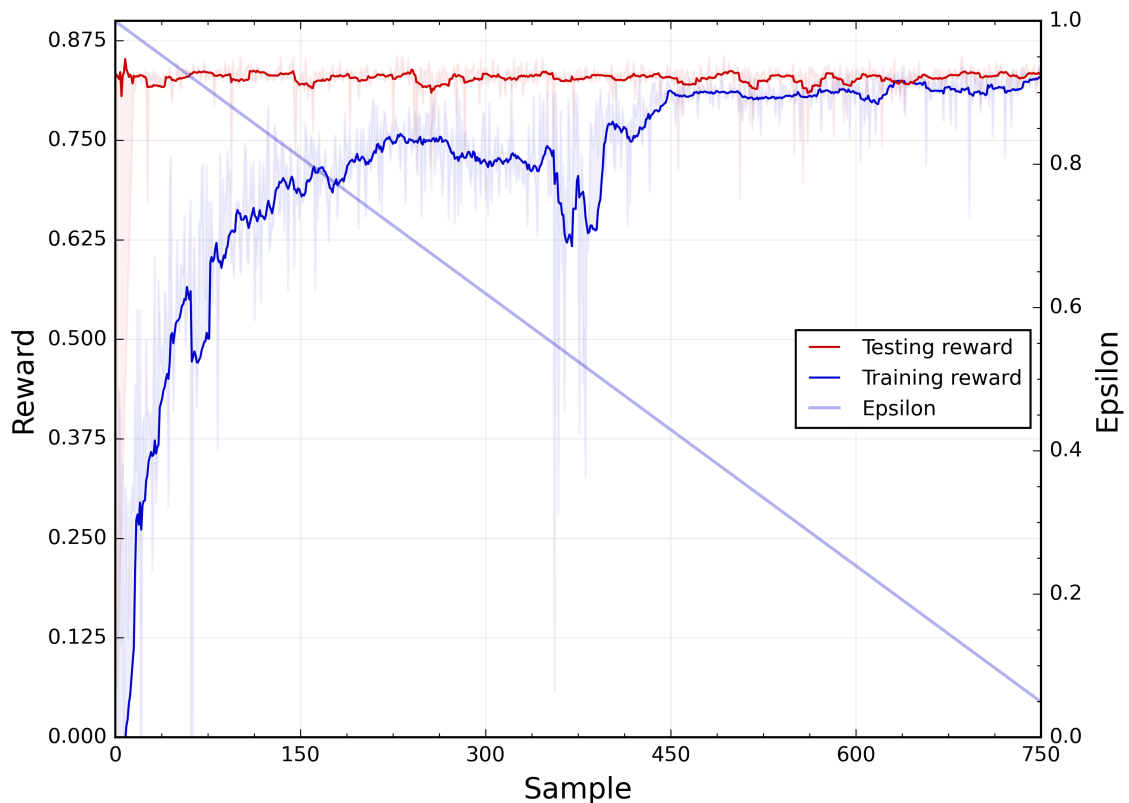
Multi-class		Single-class
MAE	RMSE	PCP (%)
0.2391	0.3892	93.9

**Source: Own Authorship (2022).**

An accurate thermal response is needed, as seen in Figures 19 and 20, for providing a target and driving series prediction horizon that resembles the states returned to the agent by the environment's model, with greater accuracy from less concerns for time efficiency, given that the learning of optimizations, by the controller, is a reflection of the offline labeling capabilities.

For training and measuring the loss of the offline labeling model, the goal MSE, seen in Algorithm 2, was set to 0, as an optimal regulation state for verifying the minimum RMSE that can be achieved episodically. As the episodic reward represents the approximation to the goal error and the increase in optimization speed, as an episode yields better rewards if the optimal action composition is reached in early iterations, a cumulative reward progression of each sequential episode, displayed in Figure 21, was used to present the training episodes. After the model conversion to a maximum reward, labeling was performed once again on all training samples to verify the final exploitative results. For an improved visualization of the reward convergence, a forward-in-time filtered representation was used while the original data dispersion was displayed with related colors in the background of each series.

**Figure 21 – Reward evolution of the labeling model.**



**Source: Own Authorship (2022).**

The labeling model outperformed the PID controller regulation performance while stabilizing after transitioning to a more exploitative policy. Thus, all optimized samples were employed for training the controller.

The number of optimized samples, obtained directly and through the sample manufacturing process, roughly presented 5% of the original dataset as a consequence of the specificity of heavy engine load conditions and rated operation at high coolant temperatures for the adopted truck configuration. To increase the controller's training dataset, the complete actuation of the PID controller was leveraged by using the original dataset, used for the selection of samples for optimization, with the replacement of samples that were outperformed by labeled optimizations.

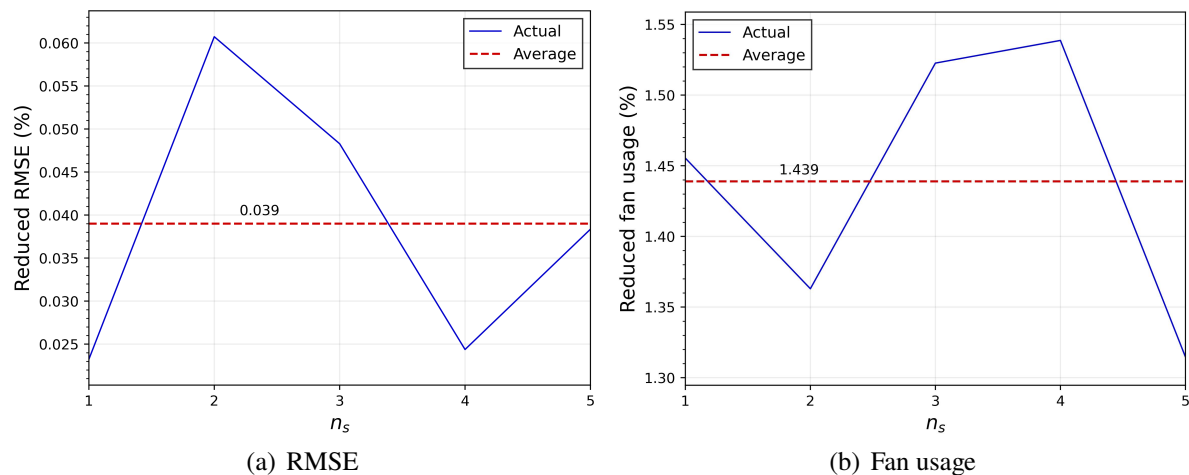
In addition to the use of mixed data, a new strategy for the evaluation of the controller model was defined as a consequence of the proportion of optimized samples and to accurately present the gains of the developed controller. Due to the learning of the observed PID controller transitions for various engine load conditions in addition to the labeled optimizations, a new testing dataset, composed of similar observations from the previously adopted training and testing sets, was acquired and specified for the controller model and for the execution of sequential inferences.



In evaluating the performance of the controller, a variable for the number of sequential runs in the test set ( $n_s$ ) was defined to verify the complete model capabilities of achieving and maintaining the temperature set point for different operating time windows. At each testing iteration, characterized by a specific value of  $n_s$ , the generated fan speed demand, resulting from sequential controller inferences, was supplied to the environment's model to accurately determine the concurrent coolant temperature states. While iterating through the test samples ( $S$ ) with  $n_s$  sequential inferences, the RMSE of the comparison between the set point and resulting target temperature states, along with the RMSE between the set point and original coolant temperature values, were collected for a comparative evaluation. Additionally, the Area Under the Curve (AUC) of both the original and generated fan speeds was calculated for a complete evaluation of the model's performance.

The number of sequential runs  $n_s$  was initially defined as 1 and, after acquiring the RMSE and AUC values for both strategies,  $n_s$  was progressively increased for the calculation of an average over a time evolution. The optimal weights of the controller model were then selected while minimizing the RMSE comparative error and fan usage. Figure 22 displays the achievements after training the MPC solution in terms of RMSE and fan usage reductions.

**Figure 22 – Results of the MPC solution.**



**Source: Own Authorship (2022).**

As seen in Figure 22, the MPC solution outperformed the PID controller with slight improvements over the maintenance of the coolant temperature and an average reduction of 1.439% in fan usage. The results expressed the need to more accurately measure the gains with the new strategy, leading to the conception of a physical, comparative and experimental evaluation.

### 4.2.3 Physical evaluation

After training, testing and validating the complete controller model and physical evaluation algorithm, the physical testing period was scheduled for a week with the performance of multiple truck runs on each test day. The proposed MPC and PID controller models were applied on sequential and intercalated runs, where, for some cases, the evaluation samples presented a high level of similarity due to a close time of day. Different ambient conditions, more specific to each testing day, heavily impacted the vehicle performance, leading to the selection of 6 runs, 3 of each strategy, based on similar external conditions.

To further diminish the impact of internal and external factors, a comparative analysis was performed based on route positions, where the minimal distance between GPS coordinates of samples from different runs defined each comparative sample.

Considering that no automation was applied for establishing equal driving conditions between runs and that some variance in internal and external conditions was unavoidable, an additional experimental strategy, for further extracting reliable samples, was developed and applied prior to the execution of the comparative evaluation.

The strategy was composed of a new TICC optimization process, where a clustered environment was sought for defining a subset of reliable samples. The dataset was built as a sequential composition of all logged test runs, and sensors were selected according to their contributions to the coolant temperature behavior. The selected sensors are presented as follows:

- Vehicle Acceleration
- Vehicle Speed
- Transmission Gear
- Engine Speed
- Road Inclination
- Ambient Temperature
- Ambient Air Pressure

To establish a metric befitting with the physical evaluation method, reliability for comparison can be weighted by the similarity that samples from different runs of the same control strategy have in terms of regulation performance. This is because of a proportional level of similarity in internal and external impacts for the comparison samples. If this similarity level is high for runs with different strategies, the contributions of different controller solutions in terms of fan usage and fuel consumption would be highlighted for comparison. Thus, the minimization of the MAD of the fuel consumption and fan usage was set as the optimization objective, where

each positional sample, presenting a common cluster in runs of the same control strategy, was selected in the search for a reliable dataset for comparison.

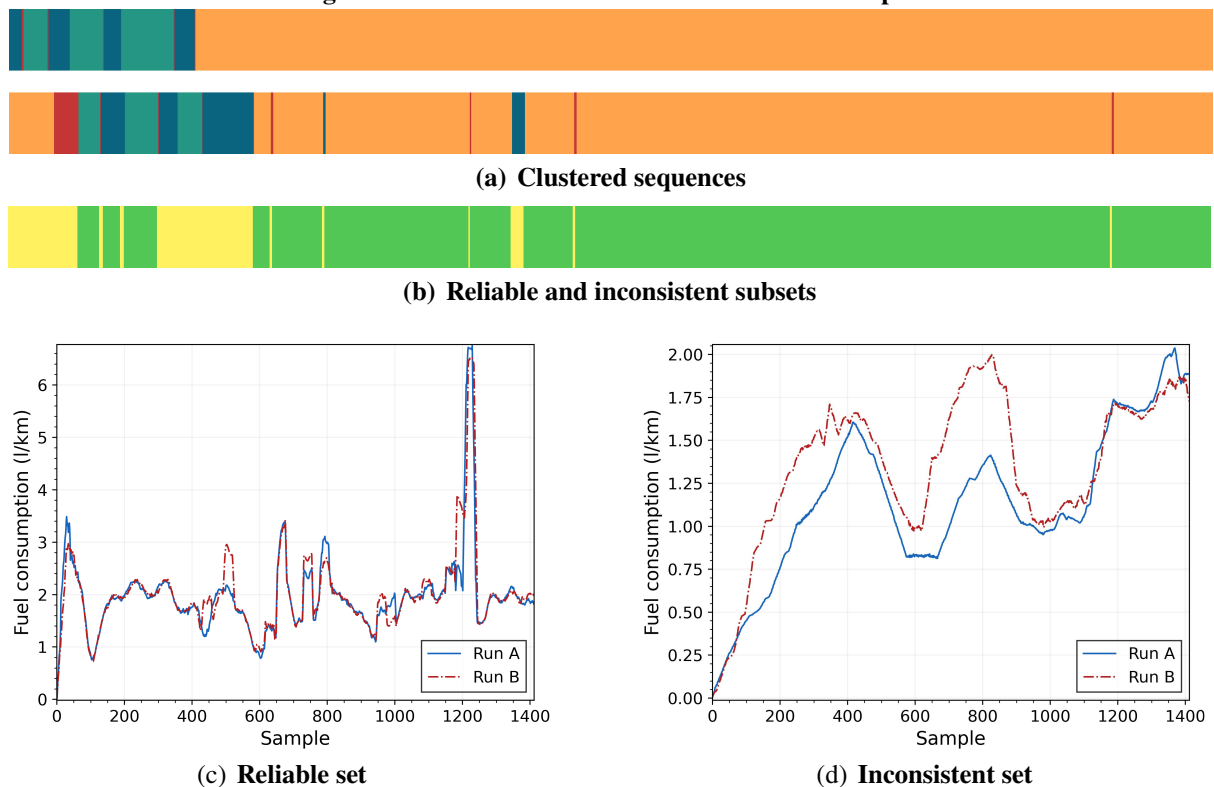
The traveled distance ( $d$ ) and fuel level ( $l$ ) signals were logged and linearized, and the fuel consumption, defined by  $c_t = \frac{|l_t - l_{t-1}|}{|d_t - d_{t-1}|}$ , was calculated and associated with each positional argument for evaluation.

The fan usage was specified as fan power due to a broader use in thermal performance evaluations, where the instantaneous values ( $p_t$ ) were obtained as a function of the logged fan speed ( $f_t$ ) with the following expression, according to the fan hardware:

$$p_t = 8727.643 - \frac{8727.652}{1 + \left(\frac{f_t}{15209.29}\right)^{3.0023}} \quad (31)$$

Figure 23 (a) shows the clustered sequences of two runs after optimization, where each common sample, collectively presented as a green subset in (b), was used for the generation of a reliable dataset for the final comparative evaluation. In (c) and (d), fuel consumption values, distributed in similar route positions for two runs of the same strategy, are presented for the reliable (green) and inconsistent (yellow) subsets, respectively. Due to the number of samples and positional arrangement, forward-in-time filtered representations were used.

**Figure 23 – Results of the extraction of reliable samples.**

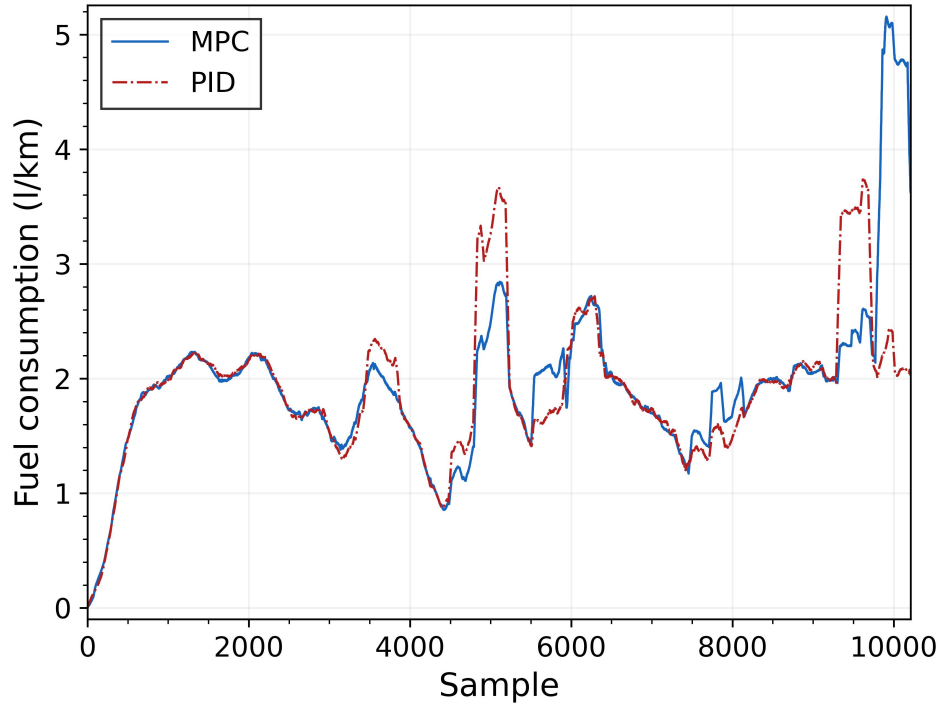


Source: Own Authorship (2022).

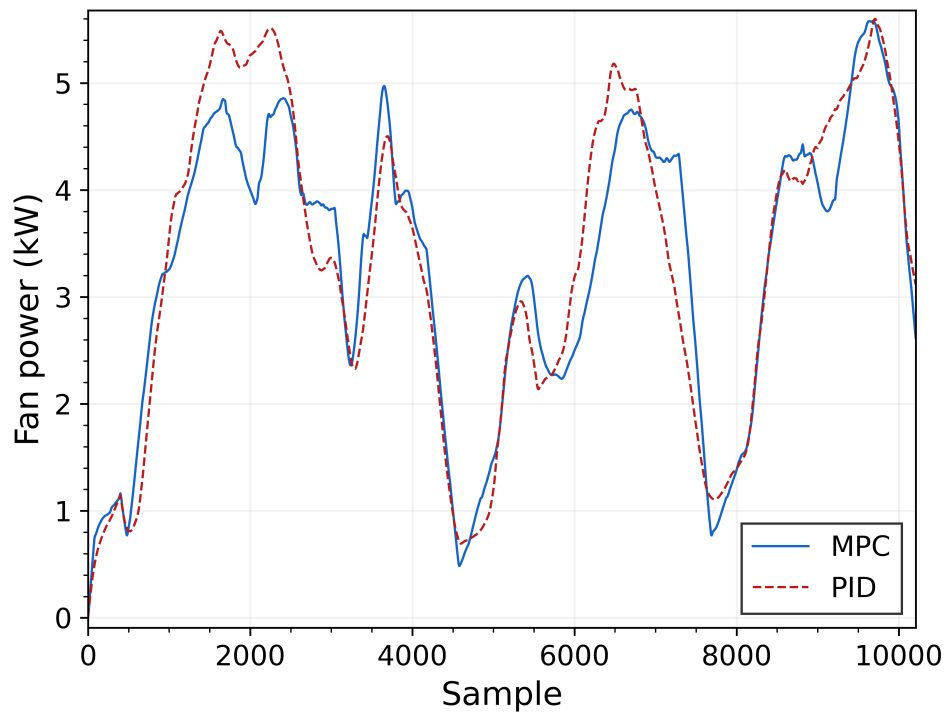
As seen in Figure 23 (a) and (b), the major portion of each original logged run was defined by a single cluster as a consequence of the imposed driving conditions, contributing to a more interpretable and consistent comparative evaluation.

With the definition of reliable samples, the datasets for comparison were built with samples that presented common cluster assignments among the 6 selected runs. The instantaneous fuel consumption and fan power values were averaged for each similar position in runs of the same strategy. Figure 24 presents the regulation performance, in terms of fuel consumption and fan power, for both strategies.

**Figure 24 – Regulation performance of the MPC and PID solutions.**



**(a) Fuel consumption**



**(b) Fan power**

**Source: Own Authorship (2022).**

An average improvement of 0.61% in the reduction of consumed fuel and 1.53% in the reduction of fan power were observed. The improvements are considered potential as specific engine load conditions were used to train, test and validate the proposed solution.

## 5 CONCLUSION

In this Master's thesis, an MPC system, consisting of offline labeling, forecasting and regulation methods, strategies and algorithms, was proposed for the thermal regulation of the refrigerant fluid. To achieve regulation optimizations solely from observations, supervised and semisupervised ML strategies were applied, improving the processed knowledge over the plant, reference and controlled signals while addressing processing time concerns with the generation of low-dimensional feature spaces. The use of an RL system, for the designed offline setting, enables the automatic search for regulation optimizations from a high-level interactive system, where simple actions are used to generate complex compositions, learned from exploring an environment of accurate target responses. A real-time operation for regulation, as inspired by the labeling setting, is achieved with the forecasting and translation of high-dimensional thermal inputs, into optimal control signals, while making use of interconnected architectures and attention mechanisms for each objective.

The challenging measurement of results from a data-driven and semisupervised controller model led to the search for experimental and comparative evaluation methods, resulting in the use of simulation and feature extraction models, and a broad range of representations. The results of the controller evaluations presented an average fan usage reduction of 1.439% with remote plant simulations and 1.53% as resulting of the physical evaluation when compared to the PID solution applied at a production scale for the adopted vehicle configuration. The improvement is a consequence of the predictive actuation of the proposed controller, learning without the specification of a range for actuation and generating a nonlinear demand of fan speeds proportional to foreseen thermal rejection states. The reduced fan usage positively impacted the fuel consumption with a reduction of 0.61%, although inconsistencies were observed for some sequences. As summarized in the evaluation of the proposed feature extraction method, regulation accuracy improvements are proportional to the description of the model dynamics, supplied to the training of ML models. Additionally, considerations for different metrics of interest, in the conception of an optimization system, would generate improvements and a more consistent thermal regulation toward fuel efficiency.

## 6 NEXT STEPS

The next steps of this work involve the design and development of a new controller version. Regarding the specification of the model dynamics, a new input treatment model will be developed for the controller and RL agent, containing the input of different vehicle characteristics relative to the thermal objective, optimally achieving a model configuration that is scalable to multiple configurations. The selection of inputs will be made according to an analysis of different vehicle configurations in production, offering support to the specifications of upcoming configurations,

The new controller model will be additionally developed as an embedded system, involving production scale concerns. The RL system of the offline labeling strategy will be built as a multi-agent solution, where different cooling agents will be explored in performing individual and collective interactions with a new simulation environment. An inner multi-agent setting for each cooling agent will be tested with a new decision-making system over the discrete action and environment spaces, comprising the magnitude of imposed gains and improvements in the action space movement for simple and complex action compositions.

The remote and physical tests and evaluations will be enhanced for multiple vehicle configurations and engine load conditions, according to the new achievements of a new version.

## REFERENCES

- ADETOLA, V.; DEHAAN, D.; GUAY, M. Adaptive model predictive control for constrained nonlinear systems. **Systems & Control Letters**, v. 58, n. 5, p. 320–326, 2009. ISSN 0167-6911. DOI: <https://doi.org/10.1016/j.sysconle.2008.12.002>. Available from: <https://www.sciencedirect.com/science/article/pii/S0167691108002120>.
- ARULKUMARAN, K. *et al.* Deep Reinforcement Learning: A Brief Survey. **IEEE Signal Processing Magazine**, v. 34, n. 6, p. 26–38, 2017.
- BEGUM, N. *et al.* Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy. In: **PROCEEDINGS of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. Sydney, NSW, Australia: Association for Computing Machinery, 2015. (KDD '15), p. 49–58. ISBN 9781450336642.
- BEMPORAD, A. Explicit Model Predictive Control. In: **Encyclopedia of Systems and Control**. Ed. by John Baillieul and Tariq Samad. London: Springer London, 2013. P. 1–9. ISBN 978-1-4471-5102-9. DOI: 10.1007/978-1-4471-5102-9\_10-1.
- BEMPORAD, A.; MORARI, M., *et al.* The Explicit Linear Quadratic Regulator for Constrained Systems. **Automatica**, Pergamon Press, Inc., USA, v. 38, n. 1, p. 3–20, Jan. 2002. ISSN 0005-1098. Available from: [https://doi.org/10.1016/S0005-1098\(01\)00174-1](https://doi.org/10.1016/S0005-1098(01)00174-1).
- BEMPORAD, A.; RICKER, L. N.; MORARI, M. **Get Started with Model Predictive Control Toolbox — mathworks.com**. [S.l.: s.n.], 2020. [https://www.mathworks.com/support/search.html?fq\[\]=asset\\_type\\_name:video&fq\[\]=category:mpc/getting-started-with-model-predictive-control-toolbox&page=1](https://www.mathworks.com/support/search.html?fq[]=asset_type_name:video&fq[]=category:mpc/getting-started-with-model-predictive-control-toolbox&page=1). [Accessed 20-Aug-2022].
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, v. 5, n. 2, p. 157–166, 1994.
- BERNDT, D. J.; CLIFFORD, J. Using Dynamic Time Warping to Find Patterns in Time Series. In: **KDD Workshop**. [S.l.: s.n.], 1994.
- BRANDES, U. A Faster Algorithm for Betweenness Centrality. **Journal of Mathematical Sociology**, v. 25, p. 163–177, 2001.
- CHISCI, L.; FALUGI, P.; ZAPPA, G. Gain-scheduling MPC of nonlinear systems. **International Journal of Robust and Nonlinear Control**, v. 13, n. 3-4, p. 295–308, 2003. DOI: <https://doi.org/10.1002/rnc.819>. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.819>.
- CHO, K.; MERRIENBOER, B. van, *et al.* **On the Properties of Neural Machine Translation: Encoder-Decoder Approaches**. CoRR, abs/1409.1259, 2014. arXiv: 1409.1259. Available from: <http://arxiv.org/abs/1409.1259>.



CHO, K.; MERRIËNBOER, B. van, *et al.* Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: **PROCEEDINGS of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. P. 1724–1734. DOI: 10.3115/v1/D14-1179. Available from: <https://aclanthology.org/D14-1179>.

DIACONESCU, E. The use of NARX neural networks to predict chaotic time series. **WSEAS Transactions on Computers archive**, v. 3, p. 182–191, 2008.

EFHEIJ, H.; ALBAGUL, A.; AMMAR ALBRAIKI, N. Comparison of Model Predictive Control and PID Controller in Real Time Process Control System. In: **2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)**. [S.l.: s.n.], 2019. P. 64–69. DOI: 10.1109/STA.2019.8717271.

ESTER, M. *et al.* A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: **PROCEEDINGS of the Second International Conference on Knowledge Discovery and Data Mining**. Portland, Oregon: AAAI Press, 1996. (KDD'96), p. 226–231.

FENG, G. A Survey on Analysis and Design of Model-Based Fuzzy Control Systems. **IEEE Transactions on Fuzzy Systems**, v. 14, n. 5, p. 676–697, 2006.

GAO, Y.; ER, M. J. NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches. **Fuzzy Sets and Systems**, Elsevier BV, v. 150, n. 2, p. 331–350, Mar. 2005.

GRUSLYS, A. *et al.* The Reactor: A fast and sample-efficient Actor-Critic agent for Reinforcement Learning. In: **INTERNATIONAL Conference on Learning Representations**. [S.l.: s.n.], 2018. Available from: <https://openreview.net/forum?id=rkHVZWZAZ>.

GU, S.; LILICRAP, T.; GHARAMANI, Z., *et al.* **Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic**. [S.l.]: arXiv, 2016. Available from: <https://arxiv.org/abs/1611.02247>.

GU, S.; LILICRAP, T.; SUTSKEVER, I., *et al.* **Continuous Deep Q-Learning with Model-based Acceleration**. [S.l.]: arXiv, 2016. Available from: <https://arxiv.org/abs/1603.00748>.

GU, S. ( *et al.* Interpolated Policy Gradient: Merging On-Policy and Off-Policy Gradient Estimation for Deep Reinforcement Learning. In: GUYON, I. *et al.* (Eds.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2017. v. 30. Available from: <https://proceedings.neurips.cc/paper/2017/file/a1d7311f2a312426d710e1c617fcbc8c-Paper.pdf>.

HALLAC, D.; NYSTRUP, P.; BOYD, S. **Greedy Gaussian Segmentation of Multivariate Time Series**. [S.l.]: arXiv, 2016. Available from: <https://arxiv.org/abs/1610.07435>.

HALLAC, D.; VARE, S., *et al.* Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data. In: **PROCEEDINGS of the 27th International Joint Conference on Artificial Intelligence**. Stockholm, Sweden: AAAI Press, 2018. (IJCAI'18), p. 5254–5258. ISBN 9780999241127.

HAUSKNECHT, M.; STONE, P. **Deep Recurrent Q-Learning for Partially Observable MDPs**. [S.l.]: arXiv, 2015. Available from: <https://arxiv.org/abs/1507.06527>.

HAYKIN, S. **Adaptive filter theory**. 4th. Upper Saddle River, NJ: Prentice Hall, 2002.

HEESS, N. *et al.* **Memory-based control with recurrent neural networks**. [S.l.]: arXiv, 2015. Available from: <https://arxiv.org/abs/1512.04455>.

HIMBERG, J. *et al.* Time series segmentation for context recognition in mobile devices. In: **PROCEEDINGS 2001 IEEE International Conference on Data Mining**. [S.l.: s.n.], 2001. P. 203–210.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-term Memory. **Neural computation**, v. 9, p. 1735–80, Dec. 1997.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society, v. 374, n. 2065, p. 20150202, Apr. 2016.

KIM, J.-S. Recent advances in adaptive MPC. In: **ICCAS 2010**. [S.l.: s.n.], 2010. P. 218–222.

KIM, J.-S.; YOON, T.-W.; SHIM, H. Switching Adaptive Output Feedback MPC for Input-constrained Neutrally Stable Linear Plants. In: **PROCEEDINGS of the 44th IEEE Conference on Decision and Control**. [S.l.: s.n.], 2005. P. 777–782.

KIM, T.-H.; SUGIE, T. Adaptive receding horizon predictive control for constrained discrete-time linear systems with parameter uncertainties. **International Journal of Control**, Taylor & Francis, v. 81, n. 1, p. 62–73, 2008. DOI: 10.1080/00207170701266779. Available from: <https://doi.org/10.1080/00207170701266779>.

KULKARNI, T. D. *et al.* **Deep Successor Reinforcement Learning**. ArXiv, abs/1606.02396, 2016.

LEE, J.; CHANG, H.-J. An application of explicit model predictive control to electric power assisted steering systems. In: **2017 11th Asian Control Conference (ASCC)**. [S.l.: s.n.], 2017. P. 2119–2124.

LILLICRAP, T. P. *et al.* Continuous control with deep reinforcement learning. In: BENGIO, Y.; LECUN, Y. (Eds.). **4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings**. [S.l.: s.n.], 2016. Available from: <http://arxiv.org/abs/1509.02971>.

LIN, T. *et al.* Learning long-term dependencies in NARX recurrent neural networks. **IEEE Transactions on Neural Networks**, v. 7, n. 6, p. 1329–1338, 1996.

MAATEN, L. van der; HINTON, G. Visualizing Data using t-SNE. **Journal of Machine Learning Research**, v. 9, n. 86, p. 2579–2605, 2008.

MAKRIDAKIS, S.; HIBON, M. ARMA Models and the Box-Jenkins Methodology. **Journal of Forecasting**, Wiley, v. 16, n. 3, p. 147–163, May 1997.

MIURA, K. An Introduction to Maximum Likelihood Estimation and Information Geometry. **Interdisciplinary Information Sciences (IIS)**, v. 17, Nov. 2011.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T., *et al.* Asynchronous Methods for Deep Reinforcement Learning. In: BALCAN, M. F.; WEINBERGER, K. Q. (Eds.). **Proceedings of The 33rd International Conference on Machine Learning**. New York, New York, USA: PMLR, June 2016. v. 48. (Proceedings of Machine Learning Research), p. 1928–1937. Available from: <https://proceedings.mlr.press/v48/mniha16.html>.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T. P., *et al.* **Asynchronous Methods for Deep Reinforcement Learning**. arXiv, 2016. Available from: <https://arxiv.org/abs/1602.01783>.

MNIH, V.; KAVUKCUOGLU, K., *et al.* Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 518, n. 7540, p. 529–533, Feb. 2015. ISSN 00280836. Available from: <http://dx.doi.org/10.1038/nature14236>.

MÜLLNER, D. **Modern hierarchical, agglomerative clustering algorithms**. [S.l.]: arXiv, 2011. Available from: <https://arxiv.org/abs/1109.2378>.

NA, S.; XUMIN, L.; YONG, G. Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. In: **2010 Third International Symposium on Intelligent Information Technology and Security Informatics**. [S.l.: s.n.], 2010. P. 63–67.

NG, A. Y.; JORDAN, M. I.; WEISS, Y. On Spectral Clustering: Analysis and an Algorithm. In: **PROCEEDINGS of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic**. Vancouver, British Columbia, Canada: MIT Press, 2001. (NIPS'01), p. 849–856.

O'DONOGHUE, B. *et al.* Combining policy gradient and Q-learning. In: **INTERNATIONAL Conference on Learning Representations**. [S.l.: s.n.], 2017. Available from: <https://openreview.net/forum?id=B1kJ6H9ex>.

OH, J. *et al.* **Control of Memory, Active Perception, and Action in Minecraft**. [S.l.]: arXiv, 2016. Available from: <https://arxiv.org/abs/1605.09128>.

PILLONETTO, G. *et al.* Kernel methods in system identification, machine learning and function estimation: A survey. **Automatica**, v. 50, n. 3, p. 657–682, 2014. ISSN 0005-1098. DOI: <https://doi.org/10.1016/j.automatica.2014.01.001>. Available from: <https://www.sciencedirect.com/science/article/pii/S000510981400020X>.

QIN, Y. *et al.* A dual-stage attention-based recurrent neural network for time series prediction. In: AAAI PRESS. **PROCEEDINGS of the 26th International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2017. P. 2627–2633. Available from: <https://www.ijcai.org/proceedings/2017/0366.pdf>.

RYBAKOV, S. *et al.* **Learning interpretable latent autoencoder representations with annotations of feature sets**. Cold Spring Harbor Laboratory, Dec. 2020. Available from: <https://doi.org/10.1101/2020.12.02.401182>.

SALEM, F.; MOSAAD, M. I. A comparison between MPC and optimal PID controllers: Case studies. In: **MICHAEL Faraday IET International Summit 2015**. [S.l.: s.n.], 2015. P. 59–65.

SCHULMAN, J. *et al.* High-Dimensional Continuous Control Using Generalized Advantage Estimation. In: **PROCEEDINGS of the International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2016.

SHOUCHE, M. *et al.* Simultaneous Constrained Model Predictive Control and Identification of DARX Processes. **Automatica**, v. 34, n. 12, p. 1521–1530, 1998. ISSN 0005-1098. DOI: [https://doi.org/10.1016/S0005-1098\(98\)80005-8](https://doi.org/10.1016/S0005-1098(98)80005-8). Available from: <https://www.sciencedirect.com/science/article/pii/S0005109898800058>.

SILVER, D. *et al.* Deterministic Policy Gradient Algorithms. In: XING, E. P.; JEBARA, T. (Eds.), 1. **Proceedings of the 31st International Conference on Machine Learning**. Beijing, China: PMLR, June 2014. v. 32. (Proceedings of Machine Learning Research, 1), p. 387–395. Available from: <https://proceedings.mlr.press/v32/silver14.html>.

SMYTH, P. Clustering Sequences with Hidden Markov Models. In: **PROCEEDINGS of the 9th International Conference on Neural Information Processing Systems**. Denver, Colorado: MIT Press, 1996. (NIPS'96), p. 648–654.

SORZANO, C. O. S.; VARGAS, J.; MONTANO, A. P. **A survey of dimensionality reduction techniques**. [S.l.]: arXiv, 2014. DOI: 10.48550/ARXIV.1403.2877. Available from: <https://arxiv.org/abs/1403.2877>.

SUN, S. *et al.* **A Survey of Optimization Methods from a Machine Learning Perspective**. [S.l.]: arXiv, 2019. DOI: 10.48550/ARXIV.1906.06821. Available from: <https://arxiv.org/abs/1906.06821>.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to Sequence Learning with Neural Networks. In: **PROCEEDINGS of the 27th International Conference on Neural Information Processing Systems - Volume 2**. Montreal, Canada: MIT Press, 2014. (NIPS'14), p. 3104–3112.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Second. [S.l.]: The MIT Press, 2018. Available from: <http://incompleteideas.net/book/the-book-2nd.html>.

SWIEF, A.; EL-ZAWAWI, A.; EL-HABROUK, M. A Survey of Model Predictive Control Development in Automotive Industries. In: **2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID)**. [S.l.: s.n.], 2019. v. 1, p. 1–7.

THARWAT, A. *et al.* Linear discriminant analysis: A detailed tutorial. **AI Communications**, IOS Press, v. 30, n. 2, p. 169–190, May 2017.

WANG, Z.; BAPST, V., *et al.* Sample Efficient Actor-Critic with Experience Replay. In: **INTERNATIONAL Conference on Learning Representations**. [S.l.: s.n.], 2017. Available from: <https://openreview.net/forum?id=HyM25Mqel>.

WANG, Z.; SCHAUL, T., *et al.* **Dueling Network Architectures for Deep Reinforcement Learning**. [S.l.]: arXiv, 2015. Available from: <https://arxiv.org/abs/1511.06581>.

WIERSTRA, D. *et al.* Recurrent policy gradients. **Logic Journal of the IGPL**, v. 18, n. 5, p. 620–634, 2010.

XU, C.; SU, Z. Identification of cell types from single-cell transcriptomes using a novel clustering method. **Bioinformatics**, v. 31, n. 12, p. 1974–1980, Feb. 2015. ISSN 1367-4803. eprint: <https://academic.oup.com/bioinformatics/article-pdf/31/12/1974/17100675/btv088.pdf>.

YAN, L.; ELGAMAL, A.; COTTRELL, G. W. Substructure Vibration NARX Neural Network Approach for Statistical Damage Inference. **Journal of Engineering Mechanics**, American Society of Civil Engineers (ASCE), v. 139, n. 6, p. 737–747, June 2013.