CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Improve Vehicle Efficiency Accuracy Through Virtual Sensors

A Comparative Study of MLP, LSTM, and Transformer Architectures with and without Physics-Informed Constraints for Fuel Consumption Prediction

Master's thesis in Computer science and engineering

Yingtian Huang
Chuyi Jiang

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Improve Vehicle Efficiency Accuracy Through Virtual Sensors

A Comparative Study of MLP, LSTM, and Transformer Architectures with and without Physics-Informed Constraints for Fuel Consumption Prediction

Yingtian Huang
Chuyi Jiang

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Improve Vehicle Efficiency Accuracy Through Virtual Sensors
A Comparative Study of MLP, LSTM, and Transformer Architectures with and without Physics-Informed Constraints for Fuel Consumption Prediction
Yingtian Huang and Chuyi Jiang

Supervisor: Mohammad Kakooei, Department of Computer Science and Engineering
Advisor: Staffan Luong, Volvo Group Trucks Technology
Vineet Kothari, Volvo Group Trucks Technology
Examiner: Ashkan Panahi, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Improve Vehicle Efficiency Accuracy Through Virtual Sensors
A Comparative Study of MLP, LSTM, and Transformer Architectures with and without Physics-Informed Constraints for Fuel Consumption Prediction
Yingtian Huang and Chuyi Jiang
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In modern heavy-duty vehicles, measuring fuel consumption relies on advanced flow meters, which are expensive and challenging to install. Volvo Group currently employs an empirical calculation model based on predefined coefficients, but this thesis explores the potential of replacing the flow meter with machine learning models. The primary objective is to develop predictive models that outperform the existing baseline. To this end, Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Transformer architectures are evaluated to determine the most suitable model for this scenario, while the impact of incorporating Physics-informed Neural Networks (PINNs) on prediction performance is also investigated. Using historical field test data, six models were trained and evaluated. All trained models demonstrated superior performance compared to the baseline. The outcomes of this thesis work have the potential to be embedded in the Electronic Control Unit (ECU) system for future field tests, providing a practical and cost-effective alternative to the physical flow meter.

Keywords: Neural network, MLP, LSTM, Transformer, PINN, fuel consumption, diesel engine, virtual sensor, digital twin.

Acknowledgements

We would like to express our sincere gratitude to our industrial supervisors, Staffan Luong and Vineet Kothari at Volvo Group, for their continuous support, guidance, and valuable discussions throughout this thesis project. Their expertise and feedback have been essential for the progress and quality of our work.

We are also grateful to our academic supervisor, Mohammad Kakooei, for providing helpful academic advice and constructive comments. Furthermore, we would like to thank our examiner, Ashkan Panahi, for insightful feedback that helped us improve this thesis.

We would like to thank Volvo Group for giving us the opportunity to conduct this project and for providing a collaborative and inspiring work environment. We are especially thankful to our colleagues in the department for their kind help, technical support, and enjoyable conversations during our time there.

Finally, we would like to thank our classmates and friends for their encouragement and our families for their constant support throughout this journey.

Chuyi Jiang & Yingtian Huang, Gothenburg, 2025-11-17

Contents

List of Figures

List of Tables

1

Introduction

This chapter provides an overview of the research project. It begins by outlining the limitations of current methods for measuring fuel consumption in background. Following this, the purpose of the proposed approach to address these limitations is introduced. The scope and limitations of the study are then discussed to clarify the boundaries of the research. Finally, relevant prior work that has inspired and informed this study is presented.

1.1 Background

With increasingly stringent EURO emission regulations being implemented in the European Union, fuel consumption has become a critical factor due to its close relationship with emissions. Moreover, fuel consumption is also a key indicator for evaluating the economic efficiency of a vehicle. A lot of effort has been taken to save fuel consumption [1].

Flow meters provide a direct approach to measuring fuel consumption and generally exhibit robust performance across various fuel types and vehicle configurations. Nonetheless, the calibration process can be intricate, and their measurement accuracy may be affected by variations in the Reynolds number of the fuel flow [2].

To address these challenges, recent research has explored the use of virtual sensors as an alternative to physical measurement devices. Virtual sensors, also referred to as soft sensors, are software-based models that estimate physical quantities using data from existing onboard sensors, such as vehicle speed, engine load, or acceleration. They enable indirect estimation of difficult-to-measure variables without additional hardware. According to [3], virtual sensors act as digital abstractions that extend the capability of physical sensors in cyber-physical systems. Their flexibility and low

cost have led to their widespread adoption in industrial process control, structural monitoring, and automotive systems [4], [5].

In the context of vehicles, virtual sensors have been proposed for estimating quantities such as engine torque, battery state-of-charge, and more recently, instantaneous fuel consumption. For example, [6] demonstrates that fuel consumption can be accurately estimated using only Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU) signals, without relying on fuel flow sensors.

Apart from direct measurement, physics-based approaches have been widely adopted in multiple areas of the automotive industry indirectly. As described in [7] and [8], fuel consumption models can be broadly categorized as white-box, gray-box, or black-box models, based on their level of physical transparency and interpretability.

However, physics-based simulations are computationally expensive and often produce a wide range of vehicle performance outputs, many of which are not directly relevant to fuel consumption estimation, resulting in inefficient use of computational resources.

As a result, it has become common practice to employ simplified, equation-based models for hybrid vehicles to support the development of control strategies. Nevertheless, these simplified models often compromise accuracy, thereby limiting their effectiveness in precisely predicting fuel consumption [9].

Given the aforementioned limitations of both direct measurement and simplified equation-based models, data-driven methods have become increasingly attractive. In studies [10]–[13] have explored various NN architectures for predicting instantaneous fuel consumption and emissions across different types of vehicles.

Building upon this, previous studies often selected NN architectures based on specific considerations, but they did not perform systematic comparisons of different models within the same project. In contrast, the present study trains multiple NN architectures - MLP, LSTM, and Transformer Encoder - on the same dataset to evaluate and compare their predictive performance. Additionally, this work investigates the potential benefits of incorporating physics-informed constraints by comparing pure data-driven models with PINNs under identical training conditions.

1.2 Purpose

The purpose of this thesis is to study machine learning methods for vehicle fuel consumption estimation. The long-term goal is to support virtual sensing as an alternative to physical flow meters. This work also plans to explore how physics knowledge can guide the training of NNs. The idea is that physics-informed models may reduce errors and give results that follow physical laws.

In the context of this project, the aim is to achieve more accurate predictions than existing embedded models. The thesis focuses on improving estimation accuracy while keeping results close to real-world ground truth.

1.3 Related Work

Three key aspects of this work are: first, the development of a NN to predict fuel consumption and other related features; second, the investigation of physical processes associated with fuel consumption; and third, the integration of physical principles into NN models. Accordingly, the literature review will primarily focus on three areas: purely data-driven NNs, physics-based modeling and simulation, and PINNs that combine physical principles with data-driven learning.

1.3.1 Data-Driven Modeling Approaches

Data-driven models leverage empirical data to learn complex relationships among features. A wide range of input features has been used in these models, including vehicle-related parameters (mass, aerodynamic and rolling resistance coefficients), powertrain characteristics (engine or motor power and torque), driving behavior metrics (speed and acceleration), and environmental factors (road grade and traffic conditions) [14]. Researchers have applied algorithms ranging from simple regressions to advanced machine learning techniques, to incorporate such diverse data. These data-driven approaches offer a flexible framework for fuel consumption estimation and driving behavior analysis, often achieving higher accuracy by learning directly from real-world data.

Early applications of machine learning to fuel consumption demonstrated the feasibility of data-driven predictions using onboard data. For example, [11] and [15] developed models using vehicle OBD/ECU sensor data to estimate both total and

instantaneous fuel consumption for heavy-duty trucks. These studies typically employed algorithms like artificial neural networks (ANN) or gradient boosting on a few key inputs such as engine speed, engine load, and vehicle speed. [15] further that the ANN achieved higher accuracy than linear regression or random forest models. Together, these works demonstrated that even with limited but well-selected features, machine learning can achieve reliable prediction performance for practical applications.

[16] provide a comprehensive review of data-driven fuel consumption prediction models. They classify approaches into traditional machine learning models, neural network-based models, and hybrid combinations of multiple techniques. In comparative experiments, more complex models achieved the best prediction performance, often reporting coefficients of determination (R^2) above 0.90 for fuel consumption estimates. Researchers also discuss that there is a trade-off between model complexity and the amount/quality of data needed: highly expressive models can yield superior accuracy if sufficient training data are available, but they may underperform or overfit with limited data.

Researchers have also explored deep learning architectures to further improve fuel consumption prediction. For instance, Fang et al. developed a convolutional neural network (CNN) based model augmented with a generative adversarial network (GAN) to capture fine-grained patterns in vehicle fuel consumption data **Fang2019**. This approach shows deep learning can model complex, nonlinear relationships in the data, which are easily missed by simpler models. Likewise, Liu et al. introduced hybrid deep neural networks and even automated machine learning strategies to optimize model architecture [17]. The results from these researches confirm that by increasing model complexity and capacity, one can better capture the complex relationship related to fuel consumption prediction.

In summary, data-driven modeling approaches for vehicle fuel consumption offer high predictive accuracy and adaptability by learning directly from data. At the same time, these approaches rely highly on the amount of data. They also tend to be less interpretable, as complex machine learning models do not inherently explain the causal impact of each input feature on fuel consumption. But data-driven methods remain a powerful tool in the vehicle dynamics modeling. And current researches continues to address their limitations.

1.3.2 Physics-Based Modeling Approaches

Physical fuel consumption models are methods based on thermodynamics, fluid mechanics, and engine subsystem dynamics, which aim to simulate the internal behavior of a combustion engine to predict fuel consumption. These models are often used for controller design, engine calibration, and virtual test benches because they provide transparency and known physics relationships.

A well-known example is a mean-value diesel engine including a Variable-Geometry Turbocharger (VGT) and Exhaust Gas Recirculation (EGR), developed in [18]. The model uses eight dynamic states (e.g., manifold pressures, turbocharger speed, actuator positions) and is optimized to capture nonlinear system dynamics while keeping computational cost moderate.

In addition to full mean-value engine models, simplified macroscopic approaches have also been studied. For instance, an average-speed-based fuel consumption model was proposed in [19], calibrated using synthetic driving cycles and transient test data. Their method segments the driving profile into microtrips and estimates the mean fuel rate based on average speed and acceleration, showing good agreement with measured values across different cycles. Although less detailed than subsystem-based models, such approaches are useful for high-level vehicle evaluation and fuel consumption prediction in traffic simulations.

Another example is a physical model of a diesel engine with a turbocharger developed for Hardware-in-the-Loop (HIL) applications in [20]. This work emphasizes not only steady-state behavior but also transient responses under varying loads and speeds, making it useful for calibration and diagnostic purposes.

In [21], a dynamic engine model aimed at energy-optimal control of heavy-duty diesel engines. A significant part of their contribution is the inclusion of air-to-fuel ratio (AFR) effects on efficiency, as well as accurate modeling of turbocharger and air/fuel path dynamics.

In heavy-duty truck applications, simplified or semi-physical models are also employed. For instance, a model combining a steady-state base with a transient correction module has been proposed for heavy-duty diesel trucks, achieving better performance than traditional VT-Micro or VT-CPFM models while maintaining lower complexity and acceptable accuracy [22].

Physical models offer many benefits. One major advantage is their interpretability: engineers can clearly understand how input states such as turbocharger speed, intake manifold pressure, or AFR influence the predicted output. These models are also inherently consistent with known physical laws, which helps ensure that predictions remain within physically realistic boundaries. For example, they do not produce negative fuel rates or violate conservation principles. Additionally, physical models can be used as reliable baselines for benchmarking or as tools to generate synthetic datasets for downstream tasks.

However, several challenges restrict their practical application. Many of these models rely on parameters that are difficult to identify or require specialized instrumentation, such as detailed pressure sensors, exhaust gas flow measurements, or manufacturer-provided turbocharger maps. Moreover, the accuracy of physical models may deteriorate under transient driving conditions if certain dynamics, like thermal lag or actuator delays, are simplified or ignored. Calibration is also non-trivial; it often demands significant engineering effort and is typically specific to a particular engine platform or vehicle configuration, reducing the transferability of the model across systems.

Reading these physical modeling works gives useful insights for combining physics with machine learning: in particular, structures for subsystem modeling (turbocharger, flow paths), dynamic states to include (manifold pressures, AFR, etc.), and methods of parameter estimation. These will inform the hybrid examined later in this thesis.

1.3.3 Physics-Informed Modeling Approaches

Physics-based models and data-driven models offer complementary benefits but also clear limits. First-principles models are physically consistent yet costly to calibrate and may miss complex real-world effects; black-box models learn rich patterns from data but can generalize poorly and lack physical guarantees. This gap motivates PINNs, which embed domain physics into learning.

A PINN is trained not only to fit data but also to satisfy governing equations by adding physics-based residuals to the loss. The concept was first introduced by in [23], where the authors demonstrated how NNs could be constrained by physical laws to solve differential equations. Later, this idea was further developed into a general framework for both forward prediction and inverse parameter estimation in [24]. Guided by physics priors, PINNs often require less data and generalize better

while ensuring physical consistency.

Researchers have applied PINNs to automotive powertrains with promising results. In [25], a diesel engine gas flow model was developed using a PINN built around a mean-value engine model with VGT and EGR systems. The network predicted key state dynamics (e.g., manifold pressures, turbo speed) from limited sensor data while also estimating hard-to-measure parameters such as EGR valve area and exhaust heat transfer coefficients. Training with both data and governing equations allowed the PINN to reconstruct internal states and identify parameter shifts that indicate faults or degradation. The model achieved high accuracy with modest data, demonstrating that PINNs can serve as effective fuel consumption models and diagnostic tools for engine health monitoring.

A more flexible approach called physics-informed data-driven modeling (PIDDM) was proposed in [14] for a plug-in hybrid electric vehicle. They noted that traditional fuel consumption models fall into three categories: physics-based, map-based, and data-driven. Each has strengths, but none performs well in all situations. To address this, they built a hybrid model combining physics and machine learning. For the engine, a dynamic engine map was used, augmented by a Transformer network to correct errors caused by fast load changes and driver behavior. For the battery, a thermal and electrical model was combined with a NN to improve predictions of temperature and state of health. This combination allowed the model to predict fuel consumption and battery state more accurately than physics-only or data-only models. Due to its accuracy and physical consistency, the model was further applied in an optimization task using a genetic algorithm to identify improved design settings for daily commuting. The resulting powertrain configuration reduced fuel consumption by approximately 4.85% and battery capacity loss by 3.7% compared to the original vehicle. This study demonstrates that integrating physical knowledge with data-driven learning can enhance prediction accuracy and support real vehicle optimization.

Another example of physics-informed modeling is the Deep Dynamics framework proposed in [26] for autonomous racing car. In this setting, the vehicle moves at very high speeds, where even small modeling errors can lead to unsafe behavior. Traditional physics-based models are difficult to tune for these extreme conditions, while purely data-driven models may violate physical rules. To address this, the authors designed a hybrid model that combines a classical vehicle dynamics model

with a NN. The physics model predicts vehicle motion based on known equations, while the NN learns unknown parameters,such as tire grip, drivetrain efficiency, and mass, that are difficult to measure directly. A key feature of their approach is a "Physics Guard" that constrains the learned parameters within reasonable physical limits. This hybrid model demonstrated strong performance, predicting the racecar's motion more accurately than a physics-only model and performing well in a real-time controller. Because the model respects physical laws, it remains stable and reliable even during aggressive driving. This study illustrates that PINNs can be effectively applied to fast and safety-critical tasks, such as autonomous racing, where both accuracy and physical consistency are essential.

Other forms of physics-informed modeling include Neural ODEs and Neural FMUs. Neural ODEs treat hidden states as continuous dynamical systems governed by differential equations, allowing the model to learn smooth time-evolving behaviors with built-in physical structure [27]. They are suitable for applications like engine dynamics or vehicle motion where time continuity matters. Neural FMUs aim to replace physics-based simulation blocks with NNs trained to match their behavior, enabling faster and more flexible system-level modeling while preserving physical consistency [28]. These approaches show that physics can be embedded not only through loss functions, as in PINNs, but also through model structure and simulation constraints.

In summary, physics-informed models offer a promising way to combine the strengths of physics-based and data-driven methods. From PINNs and their variants to Neural ODEs and Neural FMUs, these approaches improve model accuracy, generalization, and physical consistency—especially in systems like engines, powertrains, and vehicle dynamics. As these methods continue to develop, they show strong potential for use in real-time applications, diagnostics, and large-scale vehicle simulation.

1.4 Research Questions

Based on purpose and related work, the thesis is guided by the following research questions:

1. Which type of NN works best for fuel consumption estimation?
2. Can physics knowledge improve the accuracy of purely data-driven models?

2

Theory

This chapter introduces a set of fundamental concepts that form the theoretical foundation of this study. While interpretations of these concepts may vary across different contexts, the definitions and explanations presented here have been selected for their relevance and suitability to the objectives of this research. The aim is to provide a coherent theoretical framework that supports the methodology and analysis in the subsequent chapters.

2.1 Engine Structure

In this section, the structural layout of the diesel engine is introduced. A thorough understanding of the engine's working principles is important for developing an NN model, as it provides context for how the available sensor data relates to the engine's operation. Although the engine's internal mechanisms are not directly used in the implementation of this thesis work, a general understanding of its operation aids in comprehending the context and rationale behind the data and modeling choices.

The engine considered in this work is a diesel engine, as typically used in heavy-duty trucks. Unlike the spark plug ignition used in petrol engine, ignition in diesel engine occurs through mechanical compression of air, which significantly raises its temperature in cylinder.

As shown in Fig. 2.1, air is drawn in through the inlet and then compressed by a turbocharger-powered compressor, which is driven by exhaust gases. This process allows more air to enter the intake system, increasing the air density and improving engine performance.

This compression process increases the pressure of the air, which also leads to a

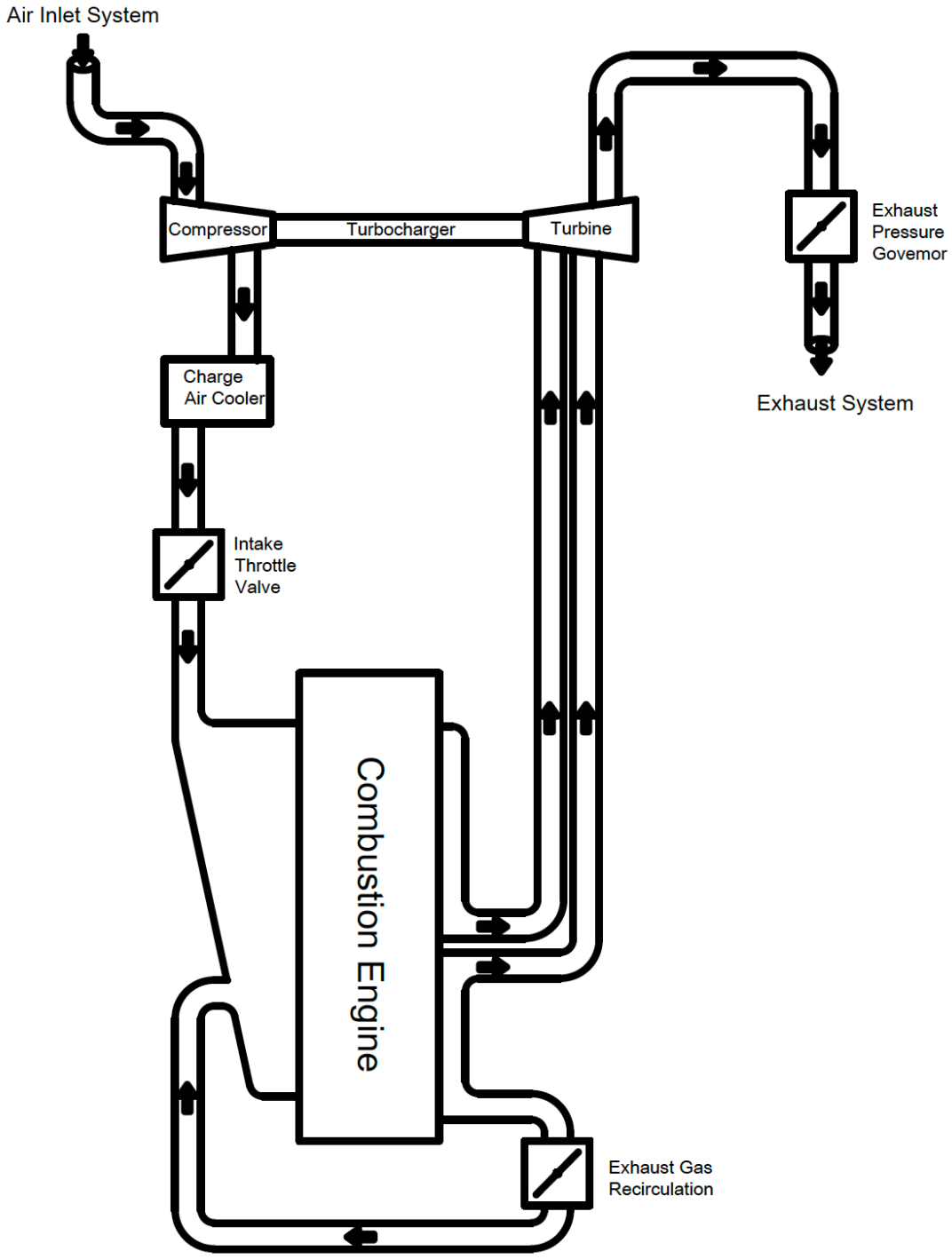


Figure 2.1: Diesel engine structure and airflow path

rise in temperature. However, hot air is less dense than cool air and contains less oxygen, which can lead to inefficiency. In order to achieve higher power in the following combustion process, an intercooler is placed between turbocharger and intake manifold.

Then the compressed air is injected into the combustion chamber. During the compression stroke, the air is compressed. The engine consists of 8 cylinders, each cylinder operates through four phases: intake, compression, combustion and exhaust. During the intake stroke, the intake valve opens and the piston moves downward, allowing fresh air to enter the cylinder. In the compression stroke, the intake valve closes. Then the piston moves upward to compress the air. The temperature will rise significantly to initiate a spontaneous ignition. Fuel is injected into cylinder filled with hot and compressed air at the end of compression process. Then a combustion follows the compression process which push the piston downward with the dramatically increases temperature and pressure. This downward force is transferred to the crankshaft as mechanical motion. During the final phase exhaustion, the exhaust valve opens and the piston moves upward again, expelling the burnt gases from cylinders.

It is common for heavy duty vehicle to use engine with 8 cylinders such as the one used in this project. The cylinders operate in a staggered sequence to ensure continuous power delivery from the combustion process, resulting in smoother engine operation and reduced vibration.

The EGR system is a structure connected to the exhaust pipe of the combustion chamber. EGR aims to reduce NOx emissions in the exhaust gas by lowering the oxygen concentration in the combustion chamber. By allowing less air into the combustion chamber, EGR lowers the peak in-cylinder temperature. NOx is primarily formed at high temperatures through the reaction between nitrogen and oxygen in the combustion air. [29]

2.2 Neural Network

NNs are computational models inspired by the structure and function of biological NNs [30]. NNs consists of a series of "neurons" which are connected units modeled brain neurons and can perform computation.

2.2.1 Network Architecture

A typical NNs consists of three different layers: input layer, one or more hidden layers and output layer [31]. Each layer has multiple neurons which performs different transformation. Neurons from different layers are connected to each other. Each neuron receives signals from the previous layer, and after transformation send the information to the next layer.

In mathematical description, for an input vector $x \in \mathbb{R}^n$ to n neurons in hidden layer, a single hidden layer transformation is defined as:

$$\begin{aligned} \mathbf{h} &= \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\ \mathbf{y} &= \mathbf{W}_2\mathbf{h} + \mathbf{b}_2 \end{aligned} \tag{2.1}$$

Where:

- n and m are input and output dimensions,
- $\mathbf{W}_1 \in \mathbb{R}^{n \times d}$ and $\mathbf{b}_1 \in \mathbb{R}^n$ are the weights and biases for the hidden layer
- $\sigma(\cdot)$ is a nonlinear activation function,
- $\mathbf{W}_2 \in \mathbb{R}^{m \times n}$ and $\mathbf{b}_2 \in \mathbb{R}^m$ are for the output layer,
- $\mathbf{y} \in \mathbb{R}^m$ is the output vector.

2.2.2 Activation Functions

NNs can approximate nonlinear equations using activation functions. There are some common activation functions that include:

- Sigmoid
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$
- Tanh
$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$
- ReLU
$$\sigma(x) = \max(0, x) \tag{2.4}$$

2.2.3 Forward and Backward Propagation

Forward propagation and backward propagation are the two core steps that enable NNs to perform prediction and training.

During forward propagation, the input data enters the input layer, passes through the hidden layer where weighted sums and activation functions are applied, and is finally transmitted to be the output layer to produce the prediction $\hat{y}$.

For backward propagation, the prediction $\hat{y}$ is compared with the ground truth y using a loss function to calculate the error. This error is then propagated backward from the output layer to the input layer using the chain rule, computing the gradients of each parameter (weights and biases). Finally, an optimizer, such as gradient descent or Adam, updates the parameters based on these gradients, allowing the NN to make more accurate predictions in the future.

2.2.4 Gradient Descent and Optimization

The parameters θ are updated iteratively to minimize the loss [32]:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L} \tag{2.5}$$

Where:

- $\eta > 0$ is the learning rate,
- $\nabla_\theta \mathcal{L}$ is the gradient of the loss with respect to θ.

Modern deep learning often uses adaptive optimizers such as Adam, RMSProp, or Adagrad for more efficient training [33].

2.2.5 Model Variants

To explore how different network architectures capture temporal and nonlinear dependencies in the data, several model variants are considered in this study. Each architecture offers distinct capabilities in representing system dynamics and learning complex input–output relationships. The following subsections describe three representative models used in this work: the MLP, the LSTM network, and the Transformer.

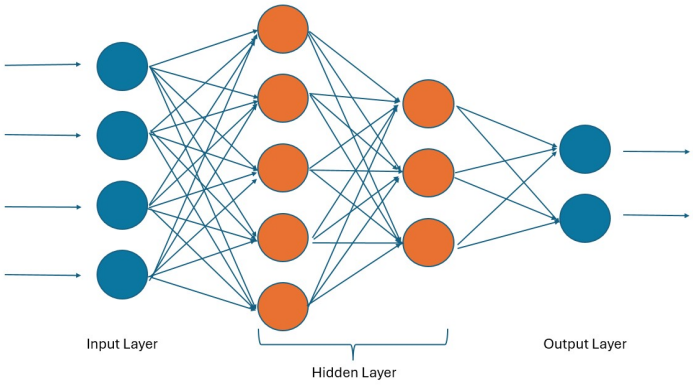


Figure 2.2: MLP Structure

2.2.5.1 Multi-Layer Perceptron

MLP is one of the fundamental architectures in ANN and serves as the basis for many advanced deep learning models. As shown in Fig. 2.2, an MLP consists of multiple layers of interconnected neurons, typically including an input layer, one or more hidden layers, and an output layer [34]. Each neuron in a layer applies a linear transformation to its inputs followed by a nonlinear activation function, enabling the network to approximate complex nonlinear mappings between input and output spaces.

Formally, given an input vector $x \in \mathbb{R}^d$, the transformation at layer l can be expressed as:

$$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right), \quad l = 1, \ldots, L, \tag{2.6}$$

where $h^{(0)} = x$, $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector of layer l, $\sigma(\cdot)$ is a nonlinear activation function (e.g., ReLU, sigmoid, or tanh), and $h^{(l)}$ is the output of layer l.

The final output of the MLP can be written as:

$$y = f_\theta(x) = W^{(L+1)} h^{(L)} + b^{(L+1)}, \tag{2.7}$$

where $\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^{L+1}$ represents the set of learnable parameters.

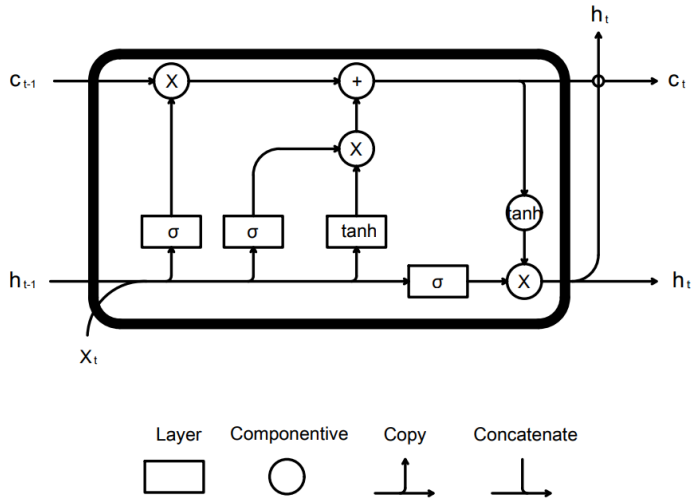


Figure 2.3: LSTM Structure

MLPs are universal function approximators, meaning that with sufficient hidden units and layers, they can approximate any continuous function to arbitrary accuracy [35], [36]. This makes them particularly effective for regression, classification, and as building blocks in more complex deep learning architectures.

2.2.5.2 Long Short-Term Memory

LSTM is a special kind of Recurrent Neural Network (RNN) which is capable of capturing long-term dependencies in sequential data [37]. Traditional RNNs suffer from the vanishing and exploding gradient problems, which hinder their ability to learn patterns across long sequences. LSTM addresses this limitation by introducing a memory cell and gating mechanisms that regulate the flow of information.

As shown in picture 2.3, an LSTM unit consists of three primary gates: the input gate, forget gate, and output gate. The input gate controls how much new information is written into the memory cell, the forget gate determines which information should be discarded from the cell, and the output gate regulates the amount of information propagated to the next hidden state. This gating structure allows LSTM networks to maintain relevant information over extended time steps and effectively model temporal dependencies. Mathematically, the LSTM operations as time step

t can be described as:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \odot \tanh(C_t) \end{aligned} \tag{2.8}$$

where x_t and h_t denote the input and hidden state at time t, C_t represents the cell state, σ is the sigmoid activation function, $\odot$ denotes element-wise multiplication, and W and b are learnable parameters. This formulation enables LSTMs to selectively remember or forget information, making them particularly effective for tasks involving time-series prediction, natural language processing, and sequence modeling.

2.2.5.3 Transformer

The Transformer architecture is a NN model designed for sequence modeling and transduction tasks, such as machine translation and time-series prediction [38]. As shown in Fig. 2.4, unlike recurrent architectures like LSTM, Transformers do not process sequences sequentially. Instead, they rely entirely on self-attention mechanisms to capture dependencies between elements at different positions in the sequences, enabling parallel computation and more efficient modeling of long-range dependencies.

A Transformer model consists of an encoder-decoder structure, where both encoder and decoder are composed of stacked layers. Each layer includes a multi-head self-attention mechanism and a position-wise feed-forward network, with residual connections and layer normalization applied to stabilized training.

The core operation, scaled dot-product attention, is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.9}$$

where Q, K, and V denotes the query, key and value matrices, respectively, and d_k is the dimension of the keys. The multi-head attention extends this by computing

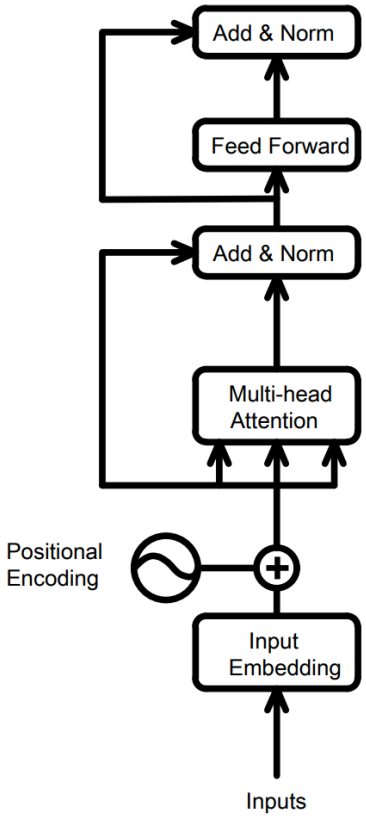


Figure 2.4: Transformer Structure

attention over multiple subspaces and concatenating the results:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.10}$$

This mechanism allows the model to focus on different parts of the sequence simultaneously, capturing complex relationship more effectively than sequential models. Additionally, positional encoding are added to input embeddings to provide information about the order of the sequence elements.

Transformers have become the foundation of many state-of-the-art models in natural language processing and other domains, owing to their ability to model long-range dependencies, facilitate parallel computation, and scale efficiently to very large database.

2.2.5.4 Physics-informed Neural Networks

PINNs are a class of deep learning model that integrate physical laws, expressed in the form of partial differential equations (PDEs) or ordinary differential equations (ODEs), into the training of NNs [39]. Unlike conventional data-driven models that rely solely on empirical data, PINNs leverage both observational data and governing equations, thereby constraining the solution space and improving generalization, especially in scenarios with limited or noisy data [40].

The core idea is to approximate the solution $u_\theta(x, t)$ of a physical system using a NN parameterized by θ. The loss function of a PINN typically combines two components: a data loss and a physics loss. The data loss enforces agreement with available measurements, while the physics loss enforces consistency with the governing equations through automatic differentiation. For example, given a PDE in the form:

$$\mathcal{N}[u(x, t)] = 0, \quad (x, t) \in \Omega, \tag{2.11}$$

where N denotes a differential operator and Ω is the domain of interest, the physics loss can be expressed as

$$\mathcal{L}_{\text{physics}} = \frac{1}{N_f} \sum_{i=1}^{N_f} |\mathcal{N}[u_\theta(x_i, t_i)]|^2 \tag{2.12}$$

The total loss then becomes

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{physics}} \tag{2.13}$$

where λ is a weighting parameter that balances the contribution of data and physics constraints.

2.3 Feature Selection Methods

In the collected dataset, a wide variety of signals are available. However, not all of them are directly related to fuel consumption, and some may only have weak or indirect correlations. To improve model efficiency and predictive performance, it is necessary to identify and retain the most relevant signals for training the machine learning models. This process is commonly addressed through feature selection, for which several established methods can be applied.

2.3.1 Machine Learning Methods

Two commonly used machine learning algorithms for feature selection are Random Forest and XGBoost. Both are tree-based models that evaluate feature importance by assessing how much each feature contributes to splitting nodes and reducing prediction error across the ensemble. Despite this similarity, they differ in their mechanisms and are suited to different scenarios.

2.3.1.1 Random Forest

When using Random Forest for feature selection, feature importance is commonly evaluated based on the reduction of impurity [41]. The impurity of a node t can be measured by Gini index, defined as

$$i(t) = 1 - \sum_{k=1}^{K} p_k^2 \tag{2.14}$$

where p_k denotes the proportion of samples of class k at node t, and K is the total number of classes. A smaller $i(t)$ indicates a purer node.

When a node t is split by feature j , the impurity reduction can be computed as:

$$\Delta i_j(t) = i(t) - \Big(p_L\, i(t_L) + p_R\, i(t_R)\Big) \tag{2.15}$$

where t_L and t_R are the left and right child nodes, and p_L and p_R denote the proportions of samples assigned to each child node. This quantity $\Delta i_j(t)$ represents the contribution of feature j at node t.

To obtain the overall importance of feature j, the impurity reductions $\Delta i_j(t)$ are summed across all nodes and all trees in the forest:

$$FI(j) = \sum_{\text{trees}} \sum_{t \in \text{splits on } j} \Delta i_j(t) \tag{2.16}$$

Finally, the feature importance scores FI of all features are normalized so that they sum to 1, producing a ranking that reflects the relative predictive power of each feature in the Random Forest model.

2.3.1.2 XGBoost

XGBoost is a boosting algorithm for gradient tree boosting [42]. It starts from an initial prediction model and sequentially adds decision trees as weak learners. At each step, the gradient and second-order gradient of the loss function with respect to the current predictions are calculated. Each new tree is then trained to fit these gradients, effectively correcting the errors of the previous model and minimizing the overall loss.

Unlike Random Forest, which evaluates feature importance based on impurity reduction, XGBoost measures the contribution of each feature using the Gain, i.e., the reduction in the loss function achieved by splitting on that feature. The Gain for each feature can be expressed as:

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma \tag{2.17}$$

where G_L, G_R are the sums of first-order gradients in the left and right child nodes, H_L, H_R are the sums of second-order gradients in the left and right child nodes, and λ, γ are regularization parameters.

A larger Gain indicates that the split using this feature results in a greater reduction of the loss function. The feature importance of a feature is then obtained by summing the Gain values across all splits in all trees, reflecting the overall contribution of the feature to the model.

2.3.2 Statistical Methods

Statistical methods evaluate the relevance of features by directly analyzing their statistical relationships with the target variable. These approaches do not rely on

training predictive models but instead exploit measures of correlation or dependency. Such methods are often computationally efficient and provide interpretable insights into the strength and type of association between input features and the output.

In this study, three commonly used statistical methods are considered: Pearson correlation coefficients, which are used to capture linear relationships between variables; Spearman coefficients, which assess monotonic but not necessarily linear dependencies; and mutual information scores, which measure more general nonlinear dependencies based on information theory. These methods offer complementary perspectives on feature relevance and can be used either as standalone criteria for feature selection or in combination with machine learning-based methods.

2.3.2.1 Pearson Correlation Coefficient

The Pearson correlation coefficient measures the strength of the linear relationship between two variables[43]. In this project, the goal is to identify which features have the greatest influence on the target variable, i.e., fuel consumption. The Pearson coefficient is denoted as r, and it is defined as:

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} \tag{2.18}$$

where:

- $\mathrm{Cov}(X,Y)$ is the covariance between X and Y.
- σ_X is the standard deviation of X
- σ_Y is the standard deviation of Y

The value of ρ lies in the range $[-1,1]$. A coefficient of $\rho = 1$ indicates a perfect positive linear correlation between the evaluated feature X and the target feature Y, $\rho = -1$ indicates a perfect negative linear correlation, and $\rho = 0$ suggests no linear correlation.

In feature selection, Pearson coefficients are calculated between each candidate feature and the target (fuel consumption), and features with coefficients below a predefined threshold can be filtered out as less relevant. This provides an efficient way to quickly eliminate features with weak correlations. However, a limitation of the

Pearson coefficient is that it can only capture linear relationships, while nonlinear dependencies remain undetected.

2.3.2.2 Spearman Rank Correlation Coefficient

Another parameter that can be considered for feature selection is the Spearman Rank Correlation Coefficient, which evaluates the monotonic relationship between two variables, not just linear correlation[44].

For two variables X and Y, the Spearman coefficient is defined as:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{2.19}$$

where $d_i = \text{rank}(X_i) - \text{rank}(Y_i)$ denotes the difference between the ranks of X_i and Y_i, and n is the number of samples.

A larger absolute value of ρ indicates a stronger monotonic relationship.

In feature selection, we calculated ρ_j for each feature X_j with respect to the target Y, and filter out features whose correlation does not meet a predefined threshold to retain only the most relevant features.

2.3.2.3 Mutual Information Scores

If the relationship between two features is nonlinear, Mutual Information (MI) can be used to evaluate their dependency. MI measures whether the increase of information in one variable can reduce the uncertainty of another variable[45]. For discrete variables X and Y, MI is defined as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2.20}$$

where $p(x,y)$ denotes the joint probability distribution of X and Y, and $p(x)$ and $p(y)$ are the marginal distributions of X and Y, respectively.

A higher $I(X;Y)$ indicates that knowing the feature X reduces more uncertainty about Y, meaning the two features are more dependent, and thus the feature will have a higher importance ranking. MI is not limited to evaluating linear relationships, but can also capture monotonic and nonlinear dependencies between variables.

3

Methods

3.1 Dataset and Preprocessing

In this study, we use a dataset collected from motorway tests to analyze the vehicle's operational and fuel consumption behavior. The dataset has been carefully preprocessed to ensure consistency, reliability, and suitability for subsequent modeling and analysis. In the following subsections, we provide a detailed description of the dataset and the preprocessing steps applied.

3.1.1 Dataset Description

The dataset comprises 21 controlled motorway test sessions carried out in Gothenburg, Sweden, between February and September 2024. All runs were executed with the same 4 x 2 heavy-duty diesel truck in identical mechanical configuration and moderate ambient conditions. Data were streamed from the vehicle CAN bus together with GPS at a fixed interval of 0.1s and stored as MATLAB `.mat` files, so each file represents a regular time series. The datasets were split into two kinds of files that deplolyed different sensor suites while keeping every other condition unchanged.

A Flex-Core GFL ground-fault sensor (in the following session called it GFL) were run in 18 test sessions. GFL is a pulse-based flow-meter. The remaining three sessions used an AVL PLUtron Coriolis flow-meter (PLUtron). PLUtron delivers and instantaneous mass-flow signal together with density and temperature. Apart from these fuel-related channels, the rest of the CAN logging configurations was identical, so the two subsets share the bast majority of variables.

The total 21 run sessions cover 1 822 kilometers and roughly 32 hours of oper-

ation. The GFL subset contributes 1 084km (20h) whereas the PLUtron subset provides 738km(12h). The original GFL files contain 125 signals and the PLUtron files 209 signals. 123 of these signals are common to both, for example, engine speed, accelerator-pedal position, and ambient temperature. The additional PLUtron-only signals include real-time fuel density, supply- and return-line pressures and the temperature-compensated mass-flow rate.

3.1.2 Data Preprocessing

The raw datasets collected from onboard sensors required several preprocessing steps to ensure that the data were clean, consistent, and ready for use in machine learning models. These steps include time synchronization, missing value handling, signal transformation, and data splitting.

3.1.2.1 Time Synchronization

Due to the distributed nature of the vehicle's sensor architecture, each signal came with its own timestamp vector. While the most signals were sampled at a nominal rate of 10Hz, few signals (e.g., the front axle speed, the compressor status) were logged at different frequencies (e.g., 50Hz, 100Hz), and others were recorded irregularly.

Prior to synchronization, any data rows with negative timestamps were discarded to avoid misalignment during resampling. Each signal was then individually resampled onto a 0.1s time period. When multiple samples fell within a time bin, their mean was taken. The signals were subsequently merged across signals using the time index. Linear interpolation was applied to fill internal missing values. Any leading missing entries were forward-filled to prevent undefined inputs at the start of each sessions.

Additionally, for signals where negative values are physically implausible (e.g., pressure, fuel flow, torque), a rectification function that sets all negative values to zero while preserving positive values.

3.1.2.2 Normalization & Data Splitting

To ensure consistent numerical ranges across all input features, the features were scaled into the range [0,1]. This approach is particularly well-suited for deep learning models, as it improves convergence stability and ensures compatibility with activation functions such as ReLU and sigmoid. Scaling parameters were computed strictly

from the training subset to prevent data leakage into validation or test phases. The normalization for a given feature value x was computed using the following equation:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

where $x_{\min}$ and $x_{\max}$ denote the minimum and maximum values of the feature in the training dataset, respectively. These scaling parameters were computed strictly from the training subset to prevent data leakage into validation or test phases.

After Normalization, the dataset was split into training, validation, and testing subsets at the session level, to prevent temporal leakage and preserve contextual continuity within each drive. Because the PLUtron dataset consists of only three sessions, we allocated one session to each subset: one for training, one for validation, and one for testing. For GFL dataset, randomly assign full sessions into the three sets using a fixed seed to ensure reproducibility, while maintaining approximate balance in driving conditions across each set.

3.1.3 Feature Selection

Feature selection is a critical step in developing accurate and interpretable predictive models. It ensures that the model focuses on the most informative signals while reducing noise, redundancy, and overfitting. In this study, we separate the discussion into target feature selection, which defines the dependent variable, and input feature selection, which identifies the independent variables used for prediction.

3.1.3.1 Target Features Selection

The target variable in this study is fuel consumption, specifically expressed as fuel flow rate. This can be reported in units such as kilograms per hour (kg/h) or grams per second (g/s), depending on the sensor configuration. Accurate target selection is critical as it directly affects the quality and interpretability of the model's predictions.

In the datasets collected, two different fuel measurement approaches were used depending on the sensor type. In the GFL-based sessions, the fuel flow is not directly measured but can be derived from a pulse-count signal. This signal represents the cumulative count of fuel injection events or flow meter pulses. From this signal, instantaneous fuel flow rate can be calculated using a calibration factor obtained

from bench testing. The *Total Pulses* variable is used as the target output when training models on GFL data.

In contrast, PLUtron-equipped sessions include direct high-accuracy measurements of the fuel flow rate, eliminating the need for signal transformation. The relevant signal, named *Volume Flow*, is provided at high resolution and is used as the target in those sessions.

Due to the difference in measurement methods, our initial plan was to treat the two signals—*Total Pulses* and *Volume Flow* from PLUtron sensor—as separate target sources and to evaluate model performance independently within each subset. This strategy was intended to ensure consistency in ground-truth definition across experimental groups while still leveraging all available data. However, after conversion and validation, the fuel flow derived from *Total Pulses* proved to be inaccurate, yielding very low agreement with reference values. Therefore, in the final setup we decided to rely exclusively on the direct measurements from the PLUtron sensor as the target variable.

In addition to this measured signal used for training, the dataset also contains a precomputed variable referred as *Fuel Rate*, which represents the output of an internal fuel consumption model embedded in the vehicle. Although the internal logic of this model is not disclosed, it is retained in our study as an external reference and used as the baseline for performance evaluation in later sections.

3.1.3.2 Input Features Selection

The selection of input features followed a structured, three-phase process that combines domain expertise, embedded model reverse-engineering, and data-driven validation. This hybrid methodology was designed to ensure that the final feature set was not only statistically robust but also physically meaningful in the context of the vehicle fuel consumption modeling. The whole process is described in Fig. 3.1.

The process began with consultations involving powertrain engineers and control system experts to gather insights into which sensor signals were most likely to influence fuel consumption. In parallel, the input list of the vehicle's baseline fuel estimation model are reviewed. Even though the model is treated as a black box, the inputs and outputs are applicable. This prior knowledge provided a physically grounded list of candidate input features, serving as a foundation for subsequent

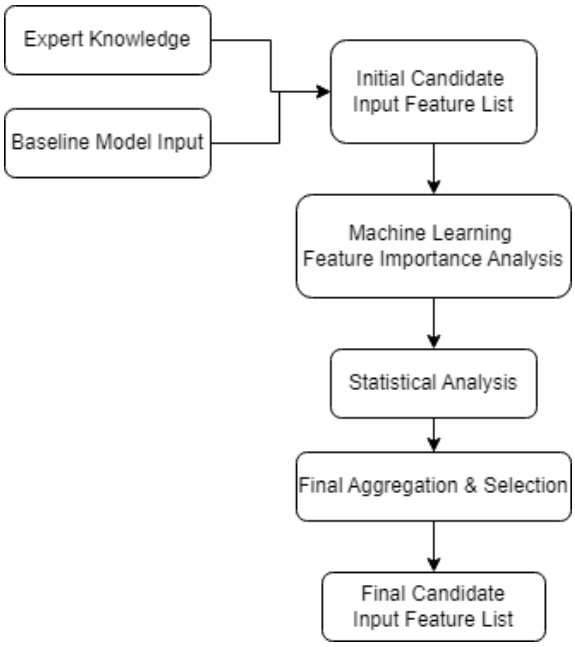


Figure 3.1: Structured input feature selection process

analysis.

The second phase involved evaluating these candidates using machine learning-based feature importance ranking. Specifically, we employed Random Forest and XGBoost models to assess the relative contribution of each feature to the prediction of the fuel rate. These models were trained using subsets of the datasets.

Lastly, statistical analysis was used to refine the feature list further. Pearson correlation coefficients, Spearman coefficient and mutual information scores were computed to detect weakly informative or highly redundant signals. Signals with consistently low correlation to the target feature or highly correlated with other features were removed unless justified by domain knowledge.

The final input feature set was determined by retaining signals that either appeared in the candidate input features list and were validated by machine learning or statistical relevance. This methodology could make sure the features have clear physical interpretability, and minimize overfitting risk.

The final set of input features used in our modeling framework is summarized in Table 3.1, grouped by physical domain for clarity. The grouping highlights the physical relevance of each variable in relation to fuel consumption.

For instance, environmental conditions such as ambient air temperature and road

inclination capture external influences that affect engine load and efficiency. Engine thermal state variables (e.g., coolant and oil temperature) represent the internal thermal balance of the powertrain, which is crucial for combustion stability and friction losses. Air and fuel system features, including boost pressure, rail pressure, and air–fuel ratio, characterize the mixture preparation and energy supply to the engine. Engine operating conditions such as engine speed and torque directly indicate the mechanical workload. Control and actuation signals (e.g., throttle position, and brake pedal position) reflect driver behavior as well as interventions from the engine control unit. Finally, emission-related and diagnostic signals (e.g., NOx levels, and oil pressure) monitor combustion quality and after treatment performance, while vehicle dynamics variables such as wheel-based vehicle speed describe the interaction between engine operation and driving conditions.

3.2 Experimental Setup

We compared three groups of models. We include the existing fuel consumption as our baseline. The second group consisted of purely data-driven NNs, namely MLP, LSTM, and TransformerEncoder. Finally, we implemented physics-informed extensions of these architectures: a PINN based on an MLP backbone, and Physics-Informed Data-Driven Models (PIDDM) with LSTM and TransformerEncoder backbones.

3.2.1 Baseline

The baseline in this study was the existing signal computed by the embedded software in the ECU namely *Fuel Rate* with unit *L/s*. Although this signal is not the most accurate measurement, it has been shown to perform well in practice and has already proven its reliability. Therefore, it provides a meaningful reference for evaluating whether the proposed models can achieve more accurate or reliable predictions.

3.2.2 Data-driven Models

We First implemented three purely data-driven NN architectures for learning the mapping from input features to fuel consumption. These models only rely on historical data without incorporating any physical knowledge or baseline signals.

- **Multilayer Perceptron (MLP)**

Table 3.1: Final input features used in the modeling framework.

Group	**Feature**	**Unit**
Environmental Conditions	Time	s
	Ambient Air Temperature	°C
	Road Inclination	%
Engine Thermal State	Engine Coolant Temperature	°C
	Engine Oil Temperature	°C
	Boost Temperature	°C
Air and Fuel System	Boost Pressure (Intake Manifold Pressure)	kPa
	Rail Pressure	bar
	Air Mass Flow	kg/s
	Fuel Value	mg/str
	Air-Fuel Ratio (AFR)	ratio
	Burned Air Fraction	%
Engine Operating Conditions	Engine Speed	rpm
	Accelerator Pedal Sensor Engine Speed	rpm
	Torque Value	Nm
	Actual Engine Percent Torque	%
Control and Actuation Signals	Engine Ignition Advance Angle	°CA
	Wastegate Position Demand	%
	Engine Cooling Fan Speed	rpm
	Throttle Position	%
	Brake Pedal Position	%
	Gear Shift State	-
	Preview Index	-
Emission-related / Diagnostic	NOx Out Level	-
	NOx In Level	-
	Oil Pressure	kPa
	Differential Pressure across Diesel Particulate Filter	-
Vehicle dynamics	Wheel Based Vehicle Speed	km/h

Implemented as a feed-forward network with several fully connected layers and ReLU activations. It maps each input sequence to a prediction without modeling temporal structure, and thus serves as the simplest reference model.

- **Long-Short Term Memory (LSTM)**
  Configured in a sequence-to-one setup, where the hidden state at the last time step is used to predict the target. This design enables the model to capture temporal in the input features.

- **Transformer Encoder**
  The Transformer Encoder is implemented as a sequence-to-one regressor. Input features at each time step are first embedded into a latent representation, then sinusoidal positional encodings are added to retain sequence order information. The embedded sequence is then processed by stacked encoder layers consisting of self-attention and feed-forward blocks.This project does not require sequence generation but a single prediction based on contextualized sequence representations, so only the encoder component is used.The final output was obtained from the hidden representation at the last time step.

3.2.3 Physics-informed Models

- **PINNs with MLP**
  The ideal way to transfer a pure MLP into a PINN is to incorporate governing equations directly into the loss function. In principle, this requires one or several governing equations that can accurately describe the engine states. However, in practice, the field test data come from a highly dynamic and nonlinear environment, making it extremely difficult to identify a governing equation that captures the system behavior using only a limited number of variables.

  Moreover, equations that are capable of describing such nonlinear dynamics usually contain a large number of unknown parameters, which are not available from the experimental setup. Therefore, inspired by [23], we adopted an alternative approach by embedding the relationships among air mass, fuel mass and air-fuel ratio (AFR) as the physics constraint in the loss function.

  Specifically, the AFR is defined as:

$$AFR(t) = \frac{m_{air}(t)}{m_{fuel}(t)} \tag{3.2}$$

To capture the dynamic coupling between air and fuel, we reformulated this algebraic definition into its differential form:

$$\frac{d}{dt}AFR(t) = \frac{\dot{m_{air}}(t)m_{fuel}(t) - m_{air}(t)\dot{m_{fuel}}(t)}{m_{fuel}(t)^2} \tag{3.3}$$

This formulation allows us to embed the temporal dynamics directly into the physics-informed loss function, which is more robust to noisy field test data and enables the NN to respect the inherent relationship among air mass, fuel mass and AFR.

While this formulation does not represent a strick physical law, but rather a mathematical relationship, it serves as a practical workaround in situations where governing equations cannot be explicitly defined. The lack of suitable governing equations is a recurring challenge in the development of PINNs, and in such scenarios, embedding domain-informed mathematical constraints can offer a feasible solution.

Con sequently, the overall loss function is extended from a purely data-driven loss to a hybrid form that combines the data loss with the additional AFR-based constraint:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{data} + \lambda_2 \mathcal{L}_{AFR} \tag{3.4}$$

where λ_1 and λ_1 are the weight coefficients that balances the contribution of the physics-informed AFR loss. This formulation enables the network to not only fit the measured data but also to remain consistent with inherent relationships among air mass, fuel mass and AFR.

- **PIDDM with LSTM**
  As introduced in the Theory section, this model is a physics-informed residual formulation, sometimes referred to as a residual LSTM. Instead of predicting the target directly, the network outputs a correction term δ, which is added to the existing baseline signal *Fuel Rate*:

$$\dot{m}_{\text{pred}} = \dot{m}_{\text{baseline}} + \Delta\dot{m}_{\text{fuel}}. \tag{3.5}$$

  The baseline acts as a physics prior, while the network learns only the systematic deviations. The training loss combines the data loss with an additional physics-informed regularization term:

$$\mathcal{L} = \text{MSE}(\dot{m}_{\text{pred}}, \dot{m}_{\text{true}}) \tag{3.6}$$

Here, $\dot{m}$ denotes the model prediction, $\dot{m}_{\text{baseline}}$ is the baseline signal provided by the ECU, $\Delta\dot{m}_{\text{fuel}}$ is the residual correction produced by the network, and the $\dot{m}_{\text{true}}$ is the ground-truth fuel consumption. And $\mathcal{L}$ represents the total loss.

- **PIDDM with Transformer Encoder**
  The Transformer-based residual model follows the same formulation as the PIDDM with LSTM: the network predicts a residual correction on top of the baseline signal, with activation and physics-informed regularization applied in the same way. The Transformer Encoder is used instead of LSTM layers to capture temporal dependencies.

3.3 Training Procedure

All models in this study were trained following a standardized training pipeline implemented in PyTorch. The procedure consists of loading data in mini-batches, forwarding inputs through the model, computing the loss, and performing parameter updates via backpropagation. Training and validation losses were monitored at each epoch.

The Adam optimizer was employed in all experiments, as it provides stable convergence. The training objective is the MSE between predicted and true fuel consumption.

Model hyperparameters (e.g., hidden size, sequence length, dropout rate) are tuned through a grid search procedure. The final configuration for each model is selected based on the best validation performance across a specific validation strategy.

To avoid overfitting, early stopping is applied. The training is terminated if no improvement is observed for a fixed number of consecutive epochs (patience threshold).

The final hyperparameter configurations for all models are summarized in Appendix Table A.1.

3.4 Evaluation Strategy

A robust evaluation strategy is crucial to ensure the reliability and generalizability of predictive models. In this study, we adopt a structured approach to model validation

and performance assessment, taking into account the limited size and time-series nature of the dataset. In the following subsections, we detail the validation strategy employed to prevent information leakage and the metrics used to quantify model performance.

3.4.1 Validation Strategy

Cross-Validation (CV) is a widely used technique for estimating generalization error and performing model selection. The core idea of CV is to divide data into separate subsets for training and validation, so that the model is always evaluated on unseen data during training. This can prevent overly optimistic estimates and reduce the risk of overfitting.

One of the most common forms is k-fold CV. In this approach, the dataset is divided into k equally sized folds. In each round, one fold is held out as the validation set while the remaining $k-1$ folds are used for training.The process is repeated k times, and the final performance is obtained by averaging across all folds.

However, this standard procedure is not suitable for this specific context. In this study, the time-series dataset consists of only three independent test sessions. This structure does not allow for a standard k-fold split across individual samples. Instead, nested cross-validation strategy was adopted. One entire session was held out as the test set, while the remaining two sessions were used for training and validation. In the fine-tuning phase, the remaining two sessions swapped between training and validation to find the best configuration. During the evaluation phase, the two sessions are merged into a single training set, and the model is retrained, then uses the test session to conclude the final performance. The process is shown in Fig. 3.2.

This strategy ensures unbiased performance estimation, prevented information leakage across sessions, and maximized the use of the limited available data.

3.4.2 Evaluation Metrics

To quantitatively assess model performance, we computed the following standard regression metrics for each fold and averaged them across all folds. Here, y_i denotes the ground-truth value and $\hat{y}_i$ the model prediction.

- **Mean Squared Error (MSE)** is the average of squared differences between

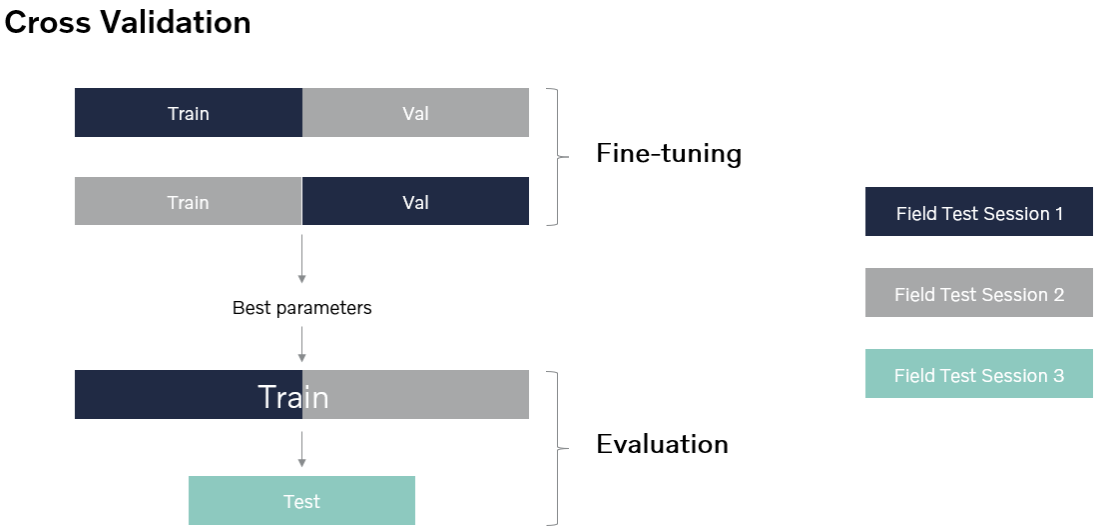


Figure 3.2: Cross-validation workflow for model training and evaluation

actual and predicted values.:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.7}$$

- **Root Mean Squared Error (RMSE)** is the square root of MSE, which preserves the unit of the target variable and is therefore more directly interpretable:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.8}$$

- **Mean Absolute Error (MAE)** calculates the average of the absolute differences between predicted and true values:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.9}$$

- **R-squared Score (R^2)** evaluates the proportion of variance in the dependent variable that is explained by the model shown below. R^2 ranges from 1 (perfect prediction) to negative values when the model performs worse than the mean predictor.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.10}$$

where $\bar{y}$ is the mean of the actual values.

All metric computations were carried out using the `scikit-learn` library. The use of multiple complementary metrics ensures a comprehensive evaluation of both the accuracy and robustness of the model predictions.

3.5 Implementation Details and Tools

The NN models were implemented using the PyTorch deep learning framework, running on Python 3.10.0. All experiments were executed on GPU-enabled remote nodes managed through a Kubernetes-based cluster, allowing dynamic allocation of hardware resources.

Code was version-controlled using Bitbucket, ensuring traceable development and experiment reproducibility. Dependency management was handled via Poetry, which provided isolated virtual environments and consistent package resolution across machines. This setup ensured that all experiments were executed under controlled and replicable software conditions.

The codebase was organized into multiple modular scripts: data preprocessing, data loading, and training logic were implemented in separate modules, while model architecture and evaluation functions were grouped within a unified script. Experimental configurations—such as batch size, learning rate, optimizer type, and cross-validation parameters—were defined directly within the training module, making it easy to control and reproduce training behaviors.

A custom logging system was implemented to record key training events, including training and validation loss values as well as evaluation metrics at each epoch. These logs were further used to generate visualizations, enabling the monitoring of training dynamics over time and facilitating early detection of overfitting, underfitting, or unstable convergence.

4

Results and Discussions

In this chapter, the results of all models are presented in both tables and figures, with a primary focus on comparative analysis. The performances of pure data-driven models and PINNs are evaluated, and particular attention is given to the model that achieves the best overall performance, for which a detailed analysis is provided.

4.1 Overview of Model Performance

In this section, we compare the performance of the baseline with all NN models. The baseline is not a learned model. Instead, it is implemented as a software module based on an engine efficiency map, which was derived from a combination of physical modeling and experimental calibration data. It serves as a reference to test whether NNs can provide more accurate and flexible predictions.

Table 4.1 presents a quantitative summary of performance metrics for the baseline and each model, with values are reported with four decimals. The metrics reported are MSE, RMSE, MAE, and R^2. Lower values in MSE, RMSE, and MAE indicate better prediction accuracy, while a higher R^2 (closer to 1) means the model explains more of the true variance. As shown in the Table 4.1, every NN model outperforms the baseline on all metrics.

4.2 Performance of Data-driven Models

We now compare the three purely data-driven models to understand their relative strengths in predicting fuel consumption. Each evaluation metric are examined to provide a more detailed analysis of model performance.

Model	MSE	RMSE	MAE	R^2
Baseline	14.2542	3.7757	1.4739	0.9537
Data-driven Models				
MLP	10.8914	3.3002	2.0465	0.9645
LSTM	0.6554	0.8096	0.4181	0.9979
Transformer	1.1632	1.0785	0.6369	0.9962
Physics-informed Models				
PINNs (MLP)	8.7944	2.9655	1.4715	0.9714
PIDDM (LSTM)	0.7649	0.8746	0.4396	0.9975
PIDDM (Transformer)	2.3325	1.5273	0.8048	0.9924

Table 4.1: Performance comparison of all models

4.2.1 Interpretation

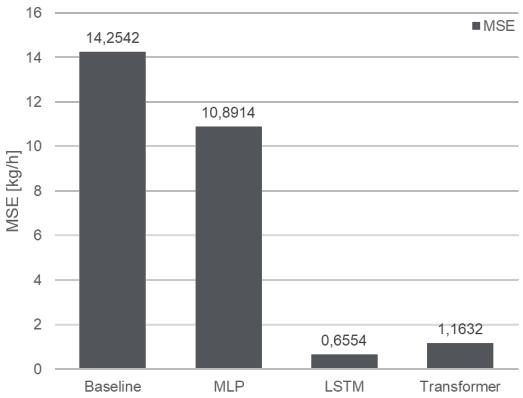
As shown in Table 4.1, the LSTM outperforms the other two models on all evaluation metrics. Fig.4.1 illustrates a comparison (as a bar chart) of the metrics achieved by each data-driven model.

In terms of R^2, both the LSTM (0.9979) and Transformer (0.9962) explain more than 99.6% of the variance in the ground truth, while the MLP is slightly lower at 0.9808. For RMSE, the LSTM again performs best with 0.8096, which is less than half of the MLP' s error (3.3002). The Transformer achieves 1.0785, which is close to the LSTM but still higher. A similar trend is seen in MAE: the LSTM has the lowest value (0.4181), the Transformer is next (0.6369), and the MLP shows the largest error (2.0465).

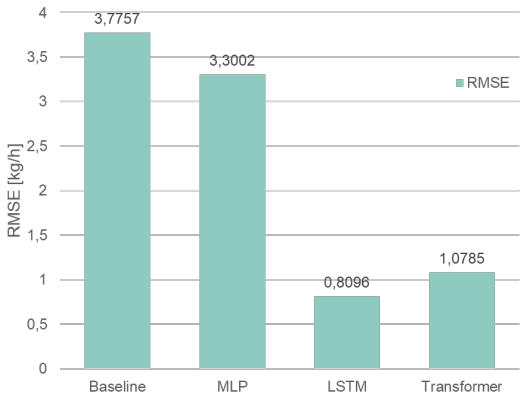
Together, these results confirm that the LSTM provides the most accurate predictions, the Transformer is a strong second, and the MLP, while weaker, still outperforms the baseline.

4.2.2 Error Analysis of MLP

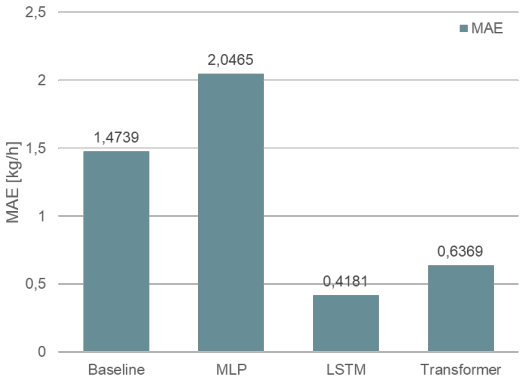
It is worth noting that MLP shows a smaller MSE and a higher R^2 compared to the baseline. However, its MAE is slightly larger. This suggests that although the MLP captures the general trend of the data better, its average absolute error is not consistently smaller. Scatter plot and residual plot can examine the prediction



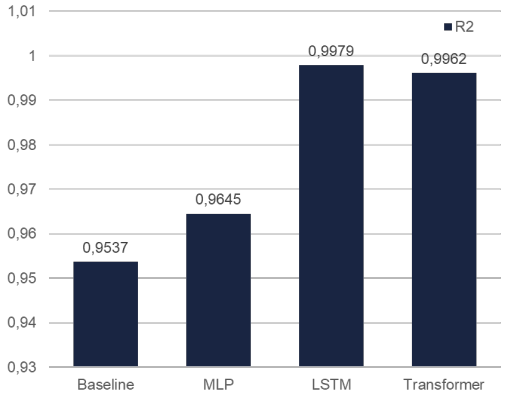
(a) RMSE comparison



(b) RMSE comparison



(c) MAE comparison



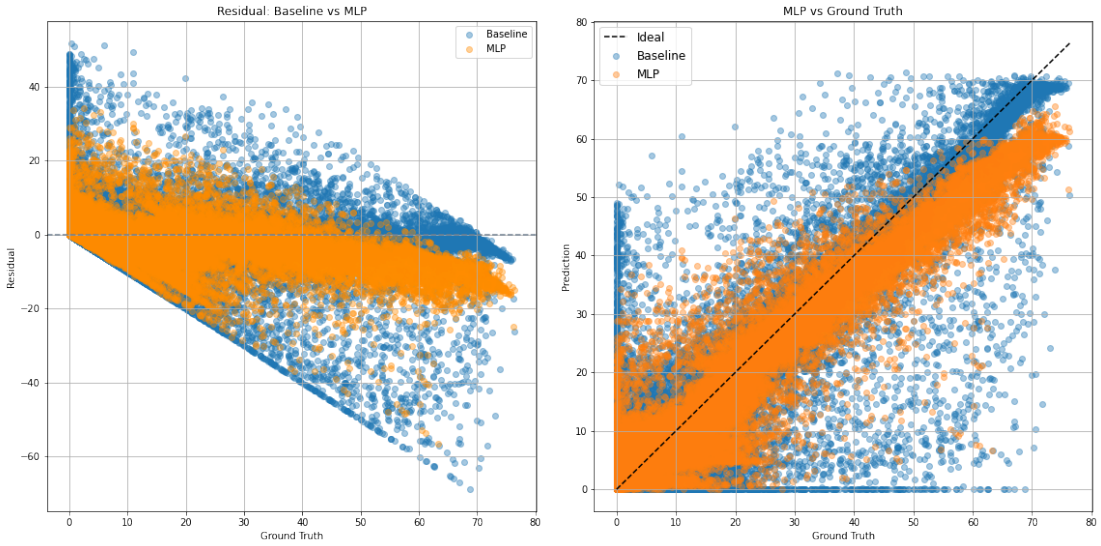
(d) R^2 comparison

Figure 4.1: Comparison of performance metrics across data-driven models

behavior.

A residual plot shows the difference between the predicted value and the ground truth on the vertical axis, and the ground truth value on the horizontal axis. This helps identify whether the prediction error changes with the value being predicted. In Fig. 4.2 (a), the baseline signal(blue) shows a wider range of errors, especially large negative residuals in high fuel consumption regions. These outliers significantly increase the MSE. In contrast, the MLP(orange) has a more compact residual distribution, with fewer extreme errors. Although it avoids large mistakes, it produces many small errors consistently, especially when the ground truth value is high. MAE is sensitive to error frequency, not just error size. This resulting in a lower MSE but a slightly higher MAE.

The scatter plot in Fig. 4.2 (b) compares predicted values (vertical axis) to ground truth (horizontal axis), and the dashed diagonal line represents ideal predictions. In this plot, the baseline model shows more dispersed points, while the MLP predictions align more closely with the diagonal line.This again confirms that MLP better captures the overall trend, therefore the R^2 is higher than baseline signal.



(a) Residual Plot

(b) Scatter Plot

Figure 4.2: Error analysis of the MLP predictions and baseline signal

4.2.3 Detailed Analysis of the LSTM Model

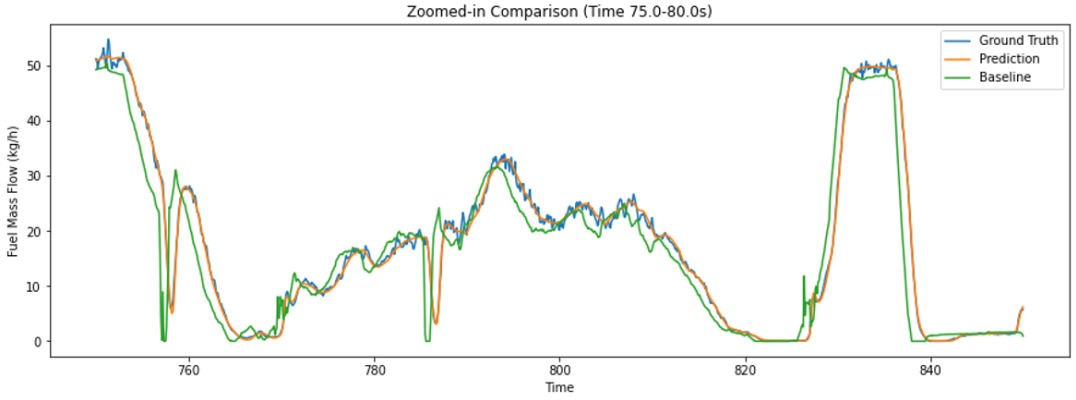
Because the LSTM achieved the best overall performance, we further examine its predictions in more detail. Fig. 4.3 shows three representative segments, where the LSTM predictions are compared with both the ground truth and the baseline signal.

In Fig. 4.3(a), a random segment illustrates a typical case. The LSTM prediction almost overlaps with the ground truth, following even the sharp rises and drops closely. By contrast, the baseline deviates substantially, especially when the fuel mass flow increases or decreases rapidly. This shows that under general conditions, the LSTM provides accurate and reliable predictions while the baseline fails to capture important variations.

Fig. 4.3(b) presents the period with the largest error. Here, the LSTM underestimates the peak magnitude but still tracks the overall shape of the curve. Although some details are missed, the LSTM remains much closer to the ground truth than the baseline, which fails to follow the true signal in this challenging case. This demonstrates that even at its weakest point, the LSTM is still robust and significantly better than the baseline.

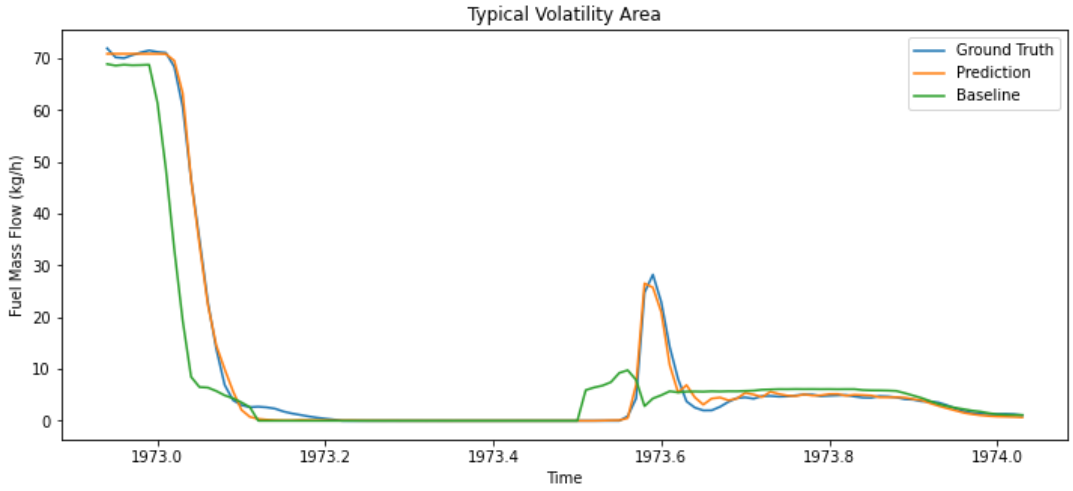
Finally, Fig. 4.3(c) shows a period with high fluctuations, where the fuel mass flow has multiple peaks and drops. The baseline cannot follow these rapid dynamics and smooths out the variations. In contrast, the LSTM prediction tracks both the sharp decreases and increases, staying very close to the ground truth. This confirms that the LSTM can handle highly dynamic scenarios, which is critical for real-world applications.

Together, these three cases highlight that the LSTM outperforms the baseline consistently in normal, difficult, and highly dynamic conditions.



(a) Random segment(Time 75.0 - 80.0s)

(b) Highest error region



(c) High fluctuation region

Figure 4.3: Predicted values of LSTM and baseline models versus ground truth

4.2.4 Discussion

All three models were trained on the same data and input features, but they differ in how they capture temporal dependencies. The MLP is a feed-forward network with no explicit sequence memory, the LSTM is a recurrent network adept at learning long-term time dependencies, and the Transformer Encoder uses self-attention mechanisms to capture both short- and long-range correlations in the sequence.

These results show the benefit of sequence modeling for this task. The LSTM can align well with changes in fuel consumption and predict peaks and drops more accurately. The Transformer also captures temporal patterns and performs close to the LSTM. Its slightly lower accuracy may come from several factors. First, the fuel consumption data is smooth and sequential, which fits the inductive bias of recurrent networks like LSTM. Second, the Transformer has higher complexity and more parameters, which makes it harder to tune and more sensitive to regularization. Third, the input length was fixed to 50 time steps in this study. This is already enough for the LSTM to capture the main dynamics, while the Transformer's strength in handling very long sequences may not give extra benefits here. As a result, the Transformer performs well but does not surpass the LSTM.

Overall, the LSTM is the most reliable pure data-driven model, with the Transformer a strong second. The MLP is weaker but still much better than the raw baseline. These results provide a reference for the next step: testing whether adding physics knowledge can improve the models further.

4.3 Performance of Physics-informed Models

Next, we assess the impact of incorporating domain physics into the NN models. Table 4.1 shows the performance of these physics-informed models compared to their purely data-driven counterparts. The effect of physics integration is model-dependent.

To visualize the effect of physics integration across models, Fig. 4.4 summarizes the RMSE performance of data-driven and physics-informed models. Since the trends of MSE, MAE, and R^2 are consistent with RMSE, only the RMSE is visualized here to avoid redundancy.

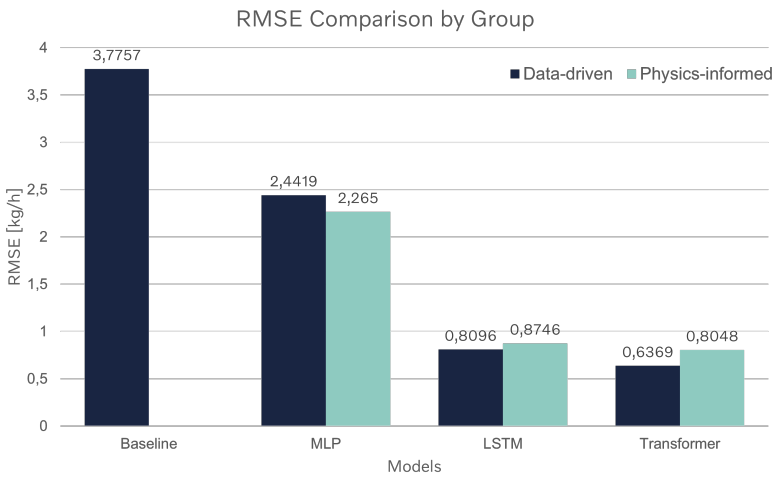


Figure 4.4: Comparison of RMSE for data-driven and physics-informed models

4.3.1 Interpretation

The MLP clearly benefits from the addition of physics: PINN-MLP achieves a lower RMSE (2.9655) than the plain MLP (3.3002) and a slightly higher R^2 (0.9714 vs 0.9645). This reduction in RMSE for the MLP after injecting physics knowledge. We observe a similar reduction in MAE (from 2.0465 down to 1.4715). These improvements, while modest, indicate that the physics-informed regularization helped the MLP correct some of its errors, likely by constraining its predictions to more physically plausible ranges and dynamics.

As illustrated in Fig. 4.5, the time series comparison between MLP and PINN-MLP reveals that PINN-MLP provides significantly closer alignment with the ground truth signal. The PINN-MLP model captures the dynamics more accurately, especially in transient regions involving rapid fuel rate changes. The physical constraints can improve the model's ability to generalize under sharp transitions and complex driving behaviors.

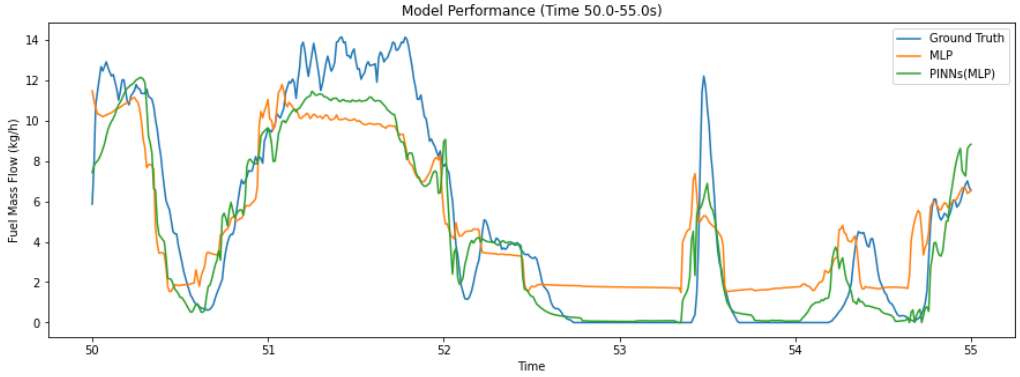


Figure 4.5: Predicted values of MLP and PINN-MLP versus ground truth

In contrast, the LSTM and Transformer did not gain accuracy from the physics-based guidance under the current implementation. The PIDDM-LSTM' s RMSE actually rose slightly to 0.8746 (from 0.8096 without physics), and its R^2 decreased marginally (0.9975 vs 0.9979). The increase in RMSE suggests that the constraints or the incorporation of the baseline signal introduced a small performance penalty for the LSTM. The effect was more pronounced for the Transformer: PIDDM-Transformer' s RMSE increased to 1.5273 from 1.0785, and R^2 dropped from 0.9962 to 0.9924.

However, when analyzing specific segments of the time series, PIDDM-LSTM occasionally aligned more closely with the ground truth. For example, during some rapid deceleration phases (see in Fig. 4.6), it produced smoother and more physically plausible predictions, potentially due to the influence of physics constraints. This suggests that PIDDM-LSTM may still offer localized benefits in capturing sharp transients or enforcing physical consistency.

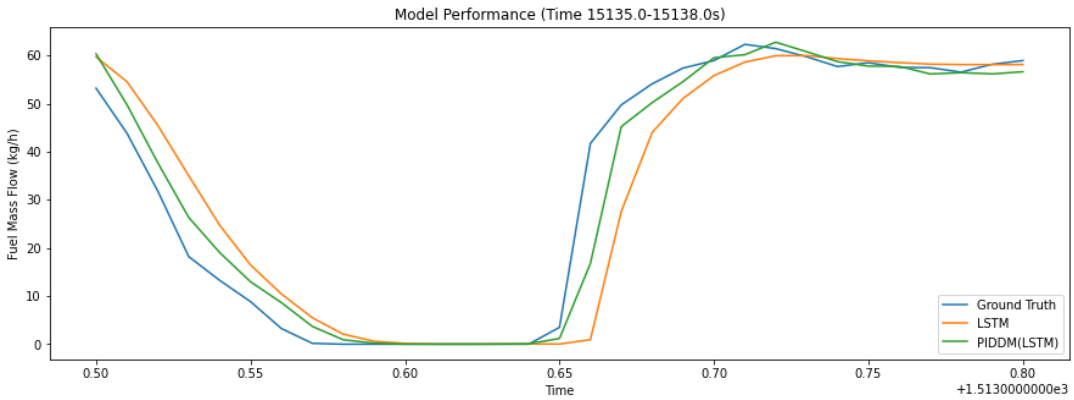


Figure 4.6: Predicted values of LSTM and PIDDM-LSTM versus ground truth

4.3.2 Discussion

The PINNs performs better than the plain MLP because of the limited capacity of the MLP in handling time-series data. As a feed-forward network, the MLP cannot directly capture temporal dependencies and is more prone to overfitting. The added physics term acts as a regularizer, reducing variance and pulling predictions back to physically plausible ranges. This explains the moderate but consistent improvement in accuracy after adding physics constraints.

However, the PINN-MLP still performs worse than the pure LSTM and Transformer models. One reason is that the physics term used here only covers part of the engine dynamics. This limited scope can guide the MLP but cannot replace the ability of more advanced models to learn complex dependencies directly from data. In addition, prior studies show that PINNs often provide strong generalization when training data is scarce. But in this work, the dataset is large, so the benefit from adding physics is reduced. As a result, the more complex data-driven models (LSTM and Transformer) already achieve better performance than the PINN-MLP.

For the LSTM and Transformer, the physics-informed versions did not improve performance. These models already captured the underlying patterns of fuel consumption, leaving little room for improvement. As baseline signal has non-negligible error and time lag, adding the baseline as a prior might have conflicted with the learned data relationships, thus leading to slight underperformance.

5

Conclusion

In this chapter, the research questions from Section 1.4 are revisited and answered. The limitations of the study are then discussed to clarify the scope of the results. Finally, directions for future work and possible real-world deployment are presented, followed by a short summary of the overall contributions of the thesis.

5.1 Answers to Research Questions

This section covers the answers to our previous research questions in Section 1.4.

Research Question 1: Which type of NN works best for fuel consumption?
Our experiments indicate that the LSTM is the best-performing architecture for estimating fuel consumption. Among the tested models and baseline signal, the LSTM achieved the highest accuracy ($R^2 \approx 0.998$ and the lowest error metrics on the validation data, outperforming both the feed-forward MLP and the Transformer Encoder.

The LSTM's strength lies in its ability to capture temporal dependencies in the engine sensor data, which proved crucial for modeling fuel consumption. In comparison, the MLP lacked sequence memory and performed less accurately, while the Transformer did well but still slightly underperformed the LSTM on this task. Therefore, in answer to RQ1, the LSTM is the most effective NN for fuel consumption prediction given our data and setup, as it provides a superior fit to the ground truth and more robust tracking of fuel consumption patterns than the other models.

Research Question 2: Can physics knowledge improve the accuracy of purely data-driven models?
In this study, integrating physics knowledge into the NNs yielded mixed results. In

one case, adding physical constraints to a simpler model (the MLP) did improve its accuracy: the PINNs achieved a modest increase in R^2 and lower error values compared to MLP. This suggests that incorporating domain physics can guide the learning process and enhance performance. In this case, the domain physics is the engine air-fuel dynamics.

However, for the more complex models (LSTM and Transformer), the inclusion of physics did not lead to better accuracy - in fact, their performance slightly declined when physics-based constraints were added. These advanced models were already capturing the essential patterns from data alone, The specific applied physics constraints may have been either redundant or restrictive, thus preventing the models from fine-tuning to the empirical data.

The answer to RQ2 is, physics knowledge can improve model accuracy under certain circumstances, particularly for simpler architectures or when the data is limited. But for more complex models, physics knowledge does not guarantee an improvement. If the physics knowledge can not perfectly aligned with the data, it may even introduce minor performance trade-offs. The effectiveness of physics integration depends on the used appropriateness of the physical model, and the capacity of the NN. This finding highlights that it's important to carefully introducing the physics into machine learning.

5.2 Limitations

Despite the results presented in this study, it is important to acknowledge several limitations that may affect the generalizability, accuracy, and applicability of the findings. These limitations arise from both the dataset and the modeling approach, and they provide context for interpreting the results and planning future work.

5.2.1 Data-related Limitations

The dataset used in this study has several limitations. In real driving conditions, many external factors affect engine dynamics and fuel use. These include weather, road type, road condition, geography, and driving behavior. Such variations were not captured in the current dataset. However, all data were collected from one truck, on a single route, and under similar weather conditions. This ensured consistency but reduced diversity. As a result, the model may not generalize well to other vehicles,

different routes, or varied environments.

Some useful sensor signals were also missing. For example, no incylinder pressure or exhaust gas composition data were available. These signals could support stronger physics-informed models with more complete engine representations. Their absence limited the scope of physical constraints in this work.

Another limitation lies in the raw signal processing. Each signal had its own timestamp, and even after synchronization, there remained small misalignments. Our preprocessing allowed the data to represent the real scenario, but we observed a slight delay between the ground-truth fuel consumption and the baseline signal. This gap influenced the training, as predictions were based on the ground truth. After consulting engineers, the preprocessing procedure was considered correct and acceptable. We therefore did not apply further corrections. This decision reflects the real impact of the dataset on the models. Interestingly, the results showed that if such adjustments were made, model predictions could improve further.

Overall, while the dataset was suitable for controlled experiments, its narrow coverage and missing signals limit both the generalization of the results and the ability to apply more advanced physics.

5.2.2 Model-related Limitations

This study also has several model-related limitations. First, we only tested two physics-informed machine learning methods. Other methods, such as Neural ODEs and their extensions, were not explored. Without broader experimentation, it is still difficult to conclude whether physics-informed methods consistently offer greater improvements than purely data-driven models.

Second, the evaluation of model performance depended on limited hyperparameter tuning. Grid search was used in this research, but the range of parameters was restricted by time and computing resources. This means that the reported results may not reflect the best possible configurations, and more extensive tuning could lead to different conclusions about model performance.

Third, deployment requirements were not fully considered during model development. This study focuses on testing methodological ideas, rather than optimizing models for real-world use. In practical deployment as a virtual sensor or in real-time prediction, computational cost, latency, and memory footprint will become critical

factors.

Overall, the models serve as a reference for the potential of physics-informed learning, but further work is needed to address their limitations.

5.2.3 Physics-related Limitations

There are also limitations in the way physics was included in this study. The applied physical knowledge only covered a part of the engine system. To obtain the most complete and accurate physical representation, one of the good approach is to build a full engine model. Such a model can take into account all relevant processes that affect fuel consumption, including combustion dynamics, heat transfer, and exhaust behavior.

Additionally, the physics implemented in this study had limited generalization. The chosen partial differential equation for AFR dynamics was designed for the given dataset and conditions, but may not directly apply to other engine types, different operating modes, or varying boundary conditions. This restricts the transferability of the approach.

Overall, while the selected physics provided useful constraints, it did not capture the full complexity of engine dynamics and may not apply well beyond this study.

5.3 Future Work

Building on the findings and limitations discussed in this study, several avenues exist for further research. Future work aims to enhance model accuracy, generalization, and applicability, particularly in real-world deployment scenarios.

5.3.1 Real-world Deployment

The ultimate goal of developing a virtual sensor is its deployment in real vehicles. This study has shown the potential of both data-driven and PINNs for fuel consumption prediction, but the evaluation has been limited to offline experiments with controlled datasets. For practical application, additional performance metrics need to be considered.

One key challenge is computational efficiency, as onboard systems usually face strict constraints in memory and processing power. Therefore, models must be designed

not only for strong offline performance but also for suitability in real-time operation.

Another important aspect of deployment is robustness across diverse real-world scenarios. Addressing this requires training on larger and more diverse datasets, incorporating a wider range of input features, and carefully balancing accuracy, generalization, and efficiency.

5.3.2 Enhanced Physics Integration

The physics used in this study provided useful guidance but remained limited in scope. Future work should focus on integrating a broader range of physical principles. Another direction is to design adaptive physics constraints that vary without being overly restricted.

5.3.3 Advanced Architectures

The architectures examined in this study mainly included MLP, LSTM and Transformer. While these provided a useful reference, there remains room to explore architectures that can more effectively capture temporal dependencies and system dynamics.

One direction is to investigate advanced recurrent structure, such as GRUs or hierarchical RNNs, which may offer a better balance between model complexity and sequence modeling ability. Another promising line of research is Neural ODEs and related continuous-time models, which align naturally with the physical nature of engine processes.

5.3.4 Online Learning and Adaptation

For real-world deployment, models must remain reliable under changing operating conditions. In practice, engine behavior evolves over time due to aging, maintenance, or variations in fuel quality, while external factors such as ambient temperature, driving style, and road environment introduce additional variability. Models trained in a static offline setting may not generalize well to these distribution shifts.

Future work should therefore explore online and continues learning methods. This enables models to adapt incrementally as new data become available. Such approaches could help the model track gradual changes without requiring complete retraining.

5.4 Summary

This thesis explored the development of NN models as a basis for a virtual fuel consumption sensor. Using real-world engine test data, several NN architectures were implemented and compared. The results showed that all tested models clearly outperformed the existing baseline, with the LSTM providing the most accurate and robust predictions. The study also examined physics-informed approaches, which proved useful in guiding simpler architectures but less effective for advanced sequence models. These findings suggest that the role of physics integration depends on model complexity and the completeness of physical knowledge. Overall, the results confirm the feasibility of building a virtual sensor for fuel consumption based on accessible on-board signals, offering a cost-effective alternative to physical sensors and a promising directions for future deployment in real vehicles.

Bibliography

[1] D. Zhu and X. Zheng, "Fuel consumption and emission characteristics in asymmetric twin-scroll turbocharged diesel engine with two exhaust gas recirculation circuits," *Applied Energy*, vol. 238, pp. 985–995, Mar. 2019, ISSN: 03062619. DOI: 10.1016/j.apenergy.2019.01.188.

[2] O. Büker, K. Stolt, C. Kroner, *et al.*, "Characterisation of a coriolis flow meter for fuel consumption measurements in realistic drive cycle tests," *Flow Measurement and Instrumentation*, vol. 93, p. 102 424, Oct. 2023, ISSN: 09555986. DOI: 10.1016/j.flowmeasinst.2023.102424.

[3] D. Martin, N. Kühl, and G. Satzger, "Virtual sensors," *Business & Information Systems Engineering*, vol. 63, pp. 315–323, 3 Jun. 2021, ISSN: 2363-7005. DOI: 10.1007/s12599-021-00689-w.

[4] L. Liu, S. M. Kuo, and M. Zhou, "Virtual sensing techniques and their applications," in *2009 International Conference on Networking, Sensing and Control*, IEEE, Mar. 2009, pp. 31–36, ISBN: 978-1-4244-3491-6. DOI: 10.1109/ICNSC.2009.4919241.

[5] N. Hirsenkorn, T. Hanke, A. Rauch, B. Dehlink, R. Rasshofer, and E. Biebl, "Virtual sensor models for real-time applications," *Advances in Radio Science*, vol. 14, pp. 31–37, Sep. 2016, ISSN: 1684-9973. DOI: 10.5194/ars-14-31-2016.

[6] F. Cellina, J. Martinez, M. Rossetti, T. Lobsiger, and R. Rudel, "Vehicle fuel consumption virtual sensing from gnss and imu measurements," *arXiv preprint arXiv:2310.01230*, 2023.

[7] M. Zhou, H. Jin, and W. Wang, "A review of vehicle fuel consumption models to evaluate eco-driving and eco-routing," *Transportation Research Part D: Transport and Environment*, vol. 49, pp. 203–218, Dec. 2016, ISSN: 13619209. DOI: 10.1016/j.trd.2016.09.008.

[8] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems*. Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-35912-5. DOI: 10.1007/978-3-642-35913-2.

[9] Z. D. Asher, D. A. Baker, and T. H. Bradley, "Prediction error applied to hybrid electric vehicle optimal fuel economy," *IEEE Transactions on Control Systems Technology*, vol. 26, pp. 2121–2134, 6 Nov. 2018, ISSN: 1063-6536. DOI: 10.1109/TCST.2017.2747502.

[10] Z. D. Asher, A. A. Galang, W. Briggs, B. Johnston, T. H. Bradley, and S. Jathar, "Economic and efficient hybrid vehicle fuel economy and emissions modeling using an artificial neural network," Apr. 2018. DOI: 10.4271/2018-01-0315.

[11] H. Abediasl, A. Ansari, V. Hosseini, C. R. Koch, and M. Shahbakhti, "Real-time vehicular fuel consumption estimation using machine learning and on-board diagnostics data," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 238, pp. 3779–3793, 12 Oct. 2024, ISSN: 0954-4070. DOI: 10.1177/09544070231185609.

[12] B. Dhanalaxmi, M. Varsha, K. R. Chowdary, and P. Mokshitha, "An enhanced fuel consumption machine learning model used in vehicles," *Journal of Physics: Conference Series*, vol. 1979, p. 012 068, 1 Aug. 2021, ISSN: 1742-6588. DOI: 10.1088/1742-6596/1979/1/012068.

[13] E. Moradi and L. Miranda-Moreno, "Vehicular fuel consumption estimation using real-world measures through cascaded machine learning modeling," *Transportation Research Part D: Transport and Environment*, vol. 88, p. 102 576, Nov. 2020, ISSN: 1361-9209. DOI: 10.1016/J.TRD.2020.102576. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S136192092030763X.

[14] N. Lei, H. Zhang, R. Li, J. Yu, H. Wang, and Z. Wang, "Physics-informed data-driven modeling approach for commuting-oriented hybrid powertrain optimization," *Energy Conversion and Management*, vol. 299, p. 117 814, Jan. 2024, ISSN: 01968904. DOI: 10.1016/j.enconman.2023.117814.

[15] S. Katreddi and A. Thiruvengadam, "Trip based modeling of fuel consumption in modern heavy-duty vehicles using artificial intelligence," *Energies*, vol. 14, no. 24, p. 8592, Dec. 2021, ISSN: 1996-1073. DOI: 10.3390/en14248592.

[16] D. Zhao, Y. Liu, J. Chen, and H. Wang, "A review of the data-driven prediction method of vehicle fuel consumption," *Energies*, vol. 16, no. 14, p. 5258, 2023. DOI: 10.3390/en16145258.

[17] X. Liu and H. Jin, "High-precision transient fuel consumption model based on support vector regression," *Fuel*, vol. 338, p. 127 368, Apr. 2023, ISSN: 00162361. DOI: 10.1016/j.fuel.2022.127368.

[18] J. Wahlström and L. Eriksson, "Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing nonlinear system dynamics," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 225, no. 7, pp. 960–986, 2011. DOI: 10.1177/0954407011398177.

[19] K. Boulouchos and et al., "Experimental calibration and validation of an average-speed fuel consumption model based on synthetic driving cycles," *SAE Technical Paper*, 2015. DOI: 10.4271/2015-01-0994.

[20] J. Yin, T. Su, Z. Guan, *et al.*, "Modeling and validation of a diesel engine with turbocharger for hardwareintheloop applications," *Energies*, vol. 10, no. 5, p. 685, 2017. DOI: 10.3390/en10050685.

[21] V. Leek *et al.*, "Developing a dynamic diesel engine model for energy optimal control," in *Proceedings of SIMS EUROSIM 2021*, 2021.

[22] R. Ding and H. Jin, "Fuel consumption model for heavy duty diesel trucks: Model development and testing," *Fuel*, 2022.

[23] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," Nov. 2017.

[24] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019, ISSN: 0021-9991. DOI: 10.1016/J.JCP.2018.10.045.

[25] K. Nath, X. Meng, D. J. Smith, and G. E. Karniadakis, "Physics-informed neural networks for predicting gas flow dynamics and unknown parameters in diesel engines," *Scientific Reports*, vol. 13, p. 13 683, 1 Aug. 2023, ISSN: 2045-2322. DOI: 10.1038/s41598-023-39989-4.

[26] J. Chrosniak, J. Ning, and M. Behl, "Deep dynamics: Vehicle dynamics modeling with a physics-constrained neural network for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 9, pp. 5292–5297, 6 Jun. 2024, ISSN: 2377-3766. DOI: 10.1109/LRA.2024.3388847.

[27] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," Dec. 2019.

[28] T. Thummerer, A. Kolesnikov, J. Gundermann, D. Ritz, and L. Mikelsons, "Paving the way for hybrid twins using neural functional mock-up units," Dec. 2023, pp. 141–150. DOI: 10.3384/ecp204141.

[29] J. Heywood, *Internal Combustion Engine Fundamentals 2E*. McGraw-Hill Education, 2019, p. 1056, ISBN: 9781260116106.

[30] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 4 Dec. 1943, ISSN: 0007-4985. DOI: 10.1007/BF02478259.

[31] M. Minsky and S. A. Papert, *Perceptrons*. The MIT Press, Sep. 2017, ISBN: 9780262343930. DOI: 10.7551/mitpress/11301.001.0001.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 6088 Oct. 1986, ISSN: 0028-0836. DOI: 10.1038/323533a0.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017.

[34] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, pp. 386–408, 6 1958, ISSN: 1939-1471. DOI: 10.1037/h0042519.

[35] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 4 Dec. 1989, ISSN: 0932-4194. DOI: 10.1007/BF02551274.

[36] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 5 Jan. 1989, ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0893608089900208.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 8 Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[38] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," Aug. 2023.

[39] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, pp. 987–1000, 5 1998, ISSN: 10459227. DOI: 10.1109/72.712178.

[40] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. Karniadakis, *Deep learning of vortex induced vibrations*, Jun. 2018. DOI: 10.31224/osf.io/fnwjy.

[41] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 1 Oct. 2001, ISSN: 0885-6125. DOI: 10.1023/A:1010933404324.

[42] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," Jun. 2016. DOI: 10.1145/2939672.2939785.

[43] K. Pearson, "Vii. mathematical contributions to the theory of evolution. –iii. regression, heredity, and panmixia," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 187, pp. 253–318, 1896. DOI: 10.1098/rsta.1896.0007.

[44] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904. DOI: 10.2307/1412159.

[45] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

A

Appendix 1

A.1 Hyperparameter Configurations and Model Complexity

This appendix lists the final configurations used for each baseline model and its corresponding physics-informed model. When the architecture is the same, the parameter count is also the same. Differences are noted explicitly.

A.1.1 MLP and PINN-MLP

Table A.1: Configurations for MLP and PINNs. PINN differs only in the output dimension (three outputs).

Parameter	**MLP**	**PINN-MLP**
Input size	28	
Hidden sizes	[128, 128, 64]	[128, 128, 64]
Dropout	0.1	0.1
Learning rate	0.005	0.005
Batch size	32	32
Epochs	50	50
Output size	1	3
parameters	28,545	28,675

A.1.2 LSTM and PIDDM–LSTM

Table A.2: Configurations for LSTM and PIDDM-LSTM. Architectures are the same; parameter counts are equal.

Parameter	**LSTM/PIDDM-LSTM**
Input size	28
Hidden size	64
Number of layers	2
Dropout	0.1
Output size	1
Sequence Length	50
Learning rate	0.01
Batch size	128
Parameters	57,409

A.1.3 Transformer Encoder and PIDDM–Transformer

Table A.3: Configurations for Transformer Encoder and PIDDM–Transformer. Architectures are the same; parameter counts are equal.

Parameter	**Transformer Encoder/PIDDM–Transformer**
Input size	28
d_{model}	128
Number of heads	4
Encoder layers	1
Dropout	0.1
Output size	1
Sequence Length	20
Learning rate	0.01
Batch size	128
Parameters	596,865