World Electric Vehicle Journal

IMPACT FACTOR 2.6

CITESCORE 5.0

Article

Holistic Approach for Automated Reverse Engineering of Unified Diagnostics Service Data

Nico Rosenberger, Nikolai Hoffmann, Alexander Mitscherlich and Markus Lienkamp

Special Issue

Electric Vehicle Technology Development, Energy and Environmental Implications, and Decarbonization: 2nd Edition

Edited by

Dr. Jinghui Wang, Dr. Hao Yang, Dr. Ran Tu and Dr. Xin He

MDPI

https://doi.org/10.3390/wevj16070384

World Electric Vehicle Journal

MDPI

Article

Holistic Approach for Automated Reverse Engineering of Unified Diagnostics Service Data

Nico Rosenberger *, Nikolai Hoffmann, Alexander Mitscherlich and Markus Lienkamp

Institute of Automotive Technology, Department of Mobility Systems Engineering, TUM School of Engineering & Design, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany
* Correspondence: nico.rosenberger@tum.de; Tel.: +49-89-289-15906

Abstract

Reverse engineering of internal vehicle communication is a crucial discipline in vehicle benchmarking. The process presents a time-consuming procedure associated with high manual effort. Car manufacturers use unique signal addresses and encodings for their internal data. Accessing this data requires either expensive tools suitable for the respective vehicles or experienced engineers who have developed individual approaches to identify specific signals. Access to the internal data enables reading the vehicle's status, and thus, reducing the need for additional test equipment. This results in vehicles closer to their production status and does not require manipulating the vehicle under study, which prevents affecting future test results. The main focus of this approach is to reduce the cost of such analysis and design a more efficient benchmarking process. In this work, we present a methodology that identifies signals without physically manipulating the vehicle. Our equipment is connected to the vehicle via the On-Board Diagnostics (OBD)-II port and uses the Unified Diagnostics Service (UDS) protocol to communicate with the vehicle. We access, capture, and analyze the vehicle's signals for future analysis. This is a holistic approach, which, in addition to decoding the signals, also grants access to the vehicle's data, which allows researchers to utilize state-of-the-art methodologies to analyze their vehicles under study by greatly reducing necessary experience, time, and cost.

Keywords: reverse engineering; signal identification; automotive ethernet; diagnostics over internet protocol; unified diagnostic service; machine learning

Academic Editors: Jinghui Wang, Hao Yang, Ran Tu and Xin He

Received: 11 June 2025
Revised: 1 July 2025
Accepted: 4 July 2025
Published: 8 July 2025

Citation: Rosenberger, N.; Hoffmann, N.; Mitscherlich, A.; Lienkamp, M. Holistic Approach for Automated Reverse Engineering of Unified Diagnostics Service Data. *World Electr. Veh. J.* **2025**, *16*, 384. https://doi.org/10.3390/wevj16070384

1. Introduction

According to [1], published in April 2024, electric vehicles accounted for 18 % of vehicle sales in 2023, rising up from 14 % in 2022 and only 2 % in 2018. This rise has been mainly led by governmental policies of restricting conventional powered vehicles [2–4] and by subsidizing purchases of electric vehicles [5]. Following these policies, original equipment manufacturers (OEMs) increased their research and development (R&D) investment in battery electric vehicle (BEV) technologies [6]. Focusing on established technologies for powertrain components, e.g., electric motors used in machine tools, new companies have entered the electric vehicle market with innovative concepts representing serious competition for OEMs. The current generation of BEVs and powertrain components illustrates the variety of different technologies applied in vehicles [7], and thus, no superior concept has been found. This was also shown in a benchmarking study where some current BEVs with different component concepts were compared on the vehicle level [8]. Identifying the most

promising electric vehicle and powertrain concept and assuming a leading position in the electric vehicle market make this a crucial time for all vehicle manufacturers.

However, investing time and money in R&D of multiple potential technologies and testing prototypes with multiple powertrain concepts does not present a target-oriented approach to the identification of superior electric vehicle concepts since these variants have already entered the market and are available. This leads to an approach of analyzing currently available vehicle and powertrain concepts and aiming to optimize R&D investments to focus on the most promising technology.

A well-known process for identifying such superior technologies is so-called benchmarking. Benchmarking is performed in multiple areas and in different approaches, with *product benchmarking* as the concept of evaluating competitors' products, investigating their concepts, and identifying their potential [9,10]. In the automotive industry, OEMs have been profiting from this process, better known as *competition analysis*, for many years to observe innovations featured in newly released vehicles by competitors.

Even though this process has been performed for a long time and OEMs have optimized this process over the years, it remains a time- and cost-consuming process. Not only does performing all targeted test procedures require high effort, but also preparing the vehicles under study for data acquisition is crucial for the following analyses. Data can either be collected by applying external sensors and manipulating the vehicle and its powertrain, or by reading the internal vehicle communication.

The widely used controller area network (CAN) bus and the emerging Automotive Ethernet standard are key technologies for communication between the electronic control units (ECUs) in modern vehicles. However, the communication is encoded with manufacturer-specific codes to prevent access to the information that is sent within the bus system [11]. Reverse engineering is the process of identifying these encodings and converting the signals from meaningless data bytes into actual physical parameters. This process, unfortunately, still remains a complex and time-consuming task [12], which requires experienced engineers and signal-specific approaches to identify the signals that are required for the respective benchmarking tests [13].

Literature research of vehicle benchmarking [14,15] confirms that relying on internal vehicle communication delivers sufficient results since both studies have been performed analyzing the CAN data of the vehicle under study, and the results provide significant information, which can, e.g., be applied in parameterizing or validating simulation models [16]. Besides the quality of the results in vehicle benchmarking, the significance of such analyses is given by qualitative results, e.g., by differences in results due to deviating boundary conditions from the official regulations.

In contrast to a detailed analysis of a single vehicle under study, when focusing on analyzing multiple vehicles, the current challenge with reverse engineering of the internal vehicle communication becomes obvious. In studies of Oh et al. [17], where 27 BEVs were included, and Grunditz et al. [18], who investigated 40 BEVs, it is shown that the analysis was limited to a small number of signals since more signals would have required more detailed reverse engineering of many different manufacturers with different encodings.

Highlighting the need for more advanced reverse engineering methods, studies [19–23] have tried to optimize the current process through automation. In these studies, several approaches have been developed to solve specific challenges by either simplifying or accelerating the manual procedures [19,20] or by implementing models for automating single modules [21–23]. The most comprehensive study is presented by Verma et al. [24], which offers a multi-step pipeline for automated reverse engineering, based on a former publication of Verma et al. [25], and concludes by providing all potential steps for reverse engineering. However, this approach still lacks one important aspect: data acquisition.

The pipeline presented within these two studies relies on already gathered data, which is a limiting factor when simplifying the reverse engineering process. To collect this data, a direct connection to the bus system is required, which can only be achieved by manipulating the vehicle under study to provide physical access to the wiring.

1.1. Contributions

Therefore, this article presents a methodology that allows researchers and engineers to fully automate the reverse engineering process. We introduce an innovative procedure that includes requesting a selection of interesting signals, capturing these signals in appropriate sampling frequencies, and identifying and converting these signals into the targeted physical parameters. Our procedure not only provides a fully comprehensive pipeline, but it also ensures the production status of the vehicle under study by communicating via the UDS protocol with our equipment connected to the vehicle's OBD-II port [26]. This, in combination with a time reduction in the whole reverse engineering process, will also enable researchers to optimize not only the process of reverse engineering but also the process of vehicle benchmarking, since vehicles can be investigated during a rental period. The main contributions of this work are summarized as follows:

- Holistic, practical, and modular approach of reverse engineering of bus data via UDS protocol communication.
- Automatic reverse engineering pipeline consisting of signal discovery, signal and ground truth data capture, and signal identification to physical meaning.
- Cost-effective application using standardized test or sensor equipment and avoiding manipulation of the vehicle under study.
- Time optimization applying efficient methods and experiment design.

1.2. Layout

This study presents the described methodology as follows: In Section 2, we introduce our concept after describing the vehicle under study and the test equipment. We explain the design of experiments for the respective parameters and the methodology for identifying and decoding the signals. In the following section, Section 3, we show the results divided into single steps through the pipeline and evaluate them against manually reverse-engineered reference signals in [27] before we discuss our achievements and open challenges later in the section. Section 4 concludes the main findings of this study and gives an outlook.

2. Materials and Methods

In this section, we introduce the vehicle under study and describe the pipeline for data access. With access to potential signals, we explain the experiment design for our ground truth data collection and the methodology for identifying the signals of interest.

2.1. Vehicle Under Study

Without loss of generality, all data gathering and analysis shown in this study was performed with a Porsche Taycan from April 2022. Like most manufacturers, Porsche produces the Taycan with multiple (optional) upgrades and additions and has already been subject to major design changes [27]. Apart from changes in design and hardware, the software of the ECUs also undergoes regular updates, which can also be distributed via over-the-air updates. The software configuration of the vehicle under study (VUS) at the time of writing is documented in Table A1.

Two features of the VUS are worth highlighting: First, it is equipped with a two-speed transmission, designed to improve powertrain efficiency across a wide range of speeds.

Second, the vehicle's chassis level is adjusted automatically to optimize air flow at higher velocities, although manual adjustment is also possible.

2.2. Pipeline for Signal Discovery

Directly accessing the data busses of the VUS without physically manipulating it, thus voiding any warranty status, was deemed infeasible. Our approach uses the OBD connector through which we establish a Unified Diagnostics Services on Internet Protocol (UDSonIP) session over Diagnostic communication over Internet Protocol (DoIP).

2.2.1. Physical Connection

ISO 13400-4 [28] defines two pin-outs for Ethernet-based communication, named Option 1 (Figure A.1 in [28]) and Option 2 (Figure B.1 in [28]), from which vehicle manufacturers may freely choose. In order to support both options, two cables with a J1962 [29] male connector on one end and an RJ45 [30] male connector on the other with a 1 m CAT5 Ethernet cable in between them were manufactured. For both cables, the RJ45 connector pin-out was manufactured according to ANSI/TIA-568A (Figure 7 in [31]), whereas the J1962 connectors conform, respectively, to Option 1 and Option 2. Deviating from ISO 13400-4, the Ethernet activation line was hardwired with an inline resistor inside the J1962 connector to Pin 16 (battery voltage). The resistor has a resistance of 500 Ω in accordance with ISO13400-3 (Table 2 in [32]). A schematic drawing of both cables is shown in Figure 1. This cable allows for the direct connection of a laptop to the VUS.

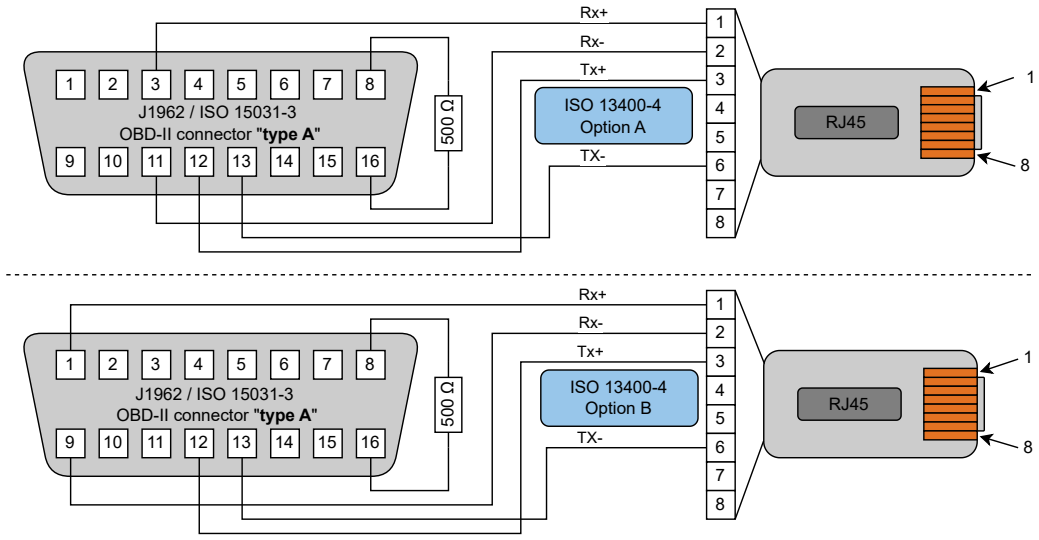

Figure 1. Pin-out for Option 1 (**top**) and Option 2 (**bottom**) looms.

2.2.2. Discover Logical Address

In order to use an UDSonIP diagnostic session, the DoIP connection as the underlying transport medium has to be established. This requires the following information: the Internet Protocol (IP) address of the VUS, the logical address of the targeted participant, in this case the Gateway of the VUS, and the logical address of the laptop.

Figure 2 shows the sequence of events that are automatically triggered by establishing a physical connection using the cable specified in Section 2.2.1 between the VUS and a laptop. Initially, both entities have to detect the presence of the physical link, after which both entities configure their network interfaces independently of each other. After some time, the VUS will broadcast three times a so-called *Vehicle Announcement Message*. This message contains the Vehicle Identification Number (VIN) and the logical address of the Gateway (Table 4 in [33]); the IP address of the Gateway can be extracted from the Ethernet

header of the broadcast message. The logical address is 16 bits wide and uniquely identifies an DoIP entity connected to the network. The range of logical addresses is divided into 13 regions, defining the expected functionality of the participant. The address space from which we may choose the logical address for the laptop starts at 0x0E80 and ends at 0x0EFF. It is reserved for external vehicle-manufacturer/aftermarket-enhanced diagnostic test equipment (Table 13 in [33]). In order to discover which addresses are available for use by the laptop, a UDS Diagnostic Session is initiated with the Gateway with alternating logical addresses until the complete range has been covered. All addresses that result in the successful creation of a diagnostic session can subsequently be used.

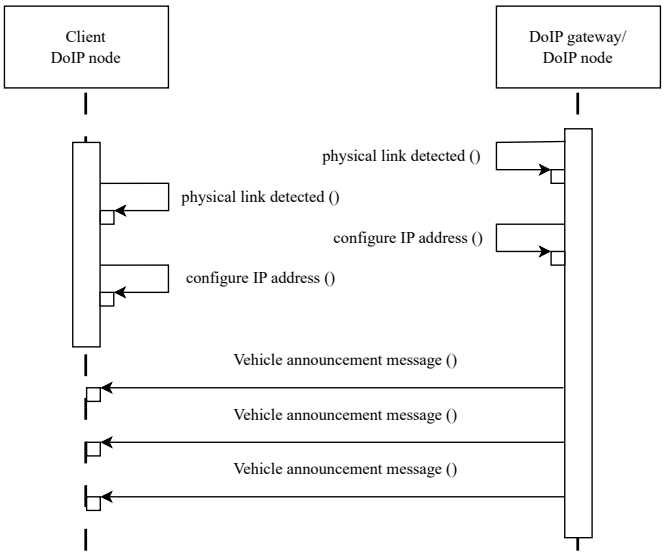

Figure 2. Vehicle announcement and identification sequence of DoIP (Figure 11 in [33]).

2.2.3. Discover Servers

In order to discover which other DoIP-enabled entities are participating in the network, a similar approach to that described in Section 2.2.2 is chosen. Using one of the available logical addresses as the source address for the laptop, the target logical address is alternated until the now complete 16-bit range is covered. All target logical addresses that result in the successful creation of a diagnostic session are addresses of the ECUs of the VUS.

2.2.4. Discover DIDs per Server

UDS defines a wide range of functions and services for communication with ECUs. For our method, only the *ReadDataByIdentifier* service is needed, which has the Service Identifier (SID) 0x22. This service is used to request parameter values identified by a 16-bit data identifier (DID).

As can be seen in Table 1, a *ReadDataByIdentifier* request message contains one or multiple DIDs; the upper limit n is defined by the OEM of the ECU. When an ECU receives a *ReadDataByIdentifier* request, it is expected to reply with either a positive response, which contains the data associated with DID, or a negative response.

The structure of a positive response message is shown in Table 2; the first byte identifies the message as a positive response for a *ReadDataByIdentifier* request, and thereafter follows the requested data. As can be seen, the payload is structured in a repeating pattern of a *dataIdentifier* followed by an arbitrary but constant (for this DID) number of associated *dataRecord* bytes.

Negative responses are identified by codes and give insight into the reason why the ECU rejected the request. Two of the five different responses are of particular interest.

Table 1. ReadDataByIdentifier request message definition (Table 186 in [26]).

Byte	Parameter Name	Byte Value
#1	ReadDataByIdentifier Request SID	22_{16}
#2	dataIdentifier[]#1 = [byte#1 (MSB)	00_{16} to FF_{16}
#3	byte#2]	00_{16} to FF_{16}
⋮	⋮	⋮
#n − 1	dataIdentifier[]#m = [byte#1 (MSB)	00_{16} to FF_{16}
#n	byte#2]	00_{16} to FF_{16}

Table 2. ReadDataByIdentifier positive response message definition (Table 188 in [26]).

Byte	Parameter Name	Byte Value
#1	ReadDataByIdentifier Response SID	62_{16}
#2	dataIdentifier[]#1 = [byte#1 (MSB)	00_{16} to FF_{16}
#3	byte#2]	00_{16} to FF_{16}
#4	dataRecord[]#1 = [data#1 …	00_{16} to FF_{16}
#(k − 1) + 4	data#k]	00_{16} to FF_{16}
⋮	⋮	⋮
#n-(o-1)-2	dataIdentifier[]#m = [byte#1 (MSB)	00_{16} to FF_{16}
#n-(o-1)-1	byte#2]	00_{16} to FF_{16}
#n-(o-1)	dataRecord[]#m = [data#1 …	00_{16} to FF_{16}
#n	data#o]	00_{16} to FF_{16}

The first negative response code of interest is 31_{16}: *requestOutOfRange*. This response can be triggered by three conditions (Table 190 in [26]):

1. None of the requested DID values are supported by the device.
2. None of the requested DIDs are supported in the current session.
3. The requested dynamic defined DID has not been assigned yet.

The other negative response code is 14_{16}: *responseTooLong*. This response is sent if the total length of the response message exceeds the limit of the underlying transport protocol, e.g., when multiple DIDs are requested in a single request (Table 190 in [26]).

In order to determine which DID values are associated with a value, every possible DID has to be tested for every ECU. This can be realized using Algorithm 1.

A more efficient approach is shown in Algorithm 2. This improved version takes advantage of the *requestOutOfRange* and *responseTooLong* behaviour. When requesting multiple DIDs at once, the server will either reply with a *requestOutOfRange* if all DIDs have no associated value. If a positive response is received, at least one DID has an associated value. And if a *responseTooLong* is received, multiple DIDs have an associated value. In these two cases, it is necessary to check every DID of the original request via separate requests. This is necessary as the length of the data values is unknown at this time, which

makes it impossible to dissect the originally received positive response. Therefore, the length of the data is saved in both cases for future use.

Algorithm 1 Sequential discovery of DIDs

```
Require: ECUs                                   ▷ array of all discovered ECUs
  while i < len(ECUs) do
    k ← 0
    while k ≤ 0xFFFF do
      response = ReadDataByIdentifier(ECUs[i],[k])
      if response is positive then
        ECUs[i].dids.append(k, len(response.value)
      else if response is requestOutOfRange then
        pass
      end if
      k ← k + 1
    end while
    i ← i + 1
  end while
```

Algorithm 2 Parallel discovery of DIDs

```
Require: ECUs                                   ▷ array of all discovered ECUs
Require: n ≥ 1                                  ▷ number of DIDs requested
  RDBI ≡ ReadDataByIdentifier
  while i < len(ECUs) do
    k ← 0
    while k < 0xFFFF do
      if k + n > 0xFFFF then
        n ← 0xFFFF − k
      end if
      dids ← [k,k + 1,...,k + n]
      response = RDBI(ECUs[i],dids)
      if response is positive then
        j ← 1
        while j ≤ n do
          response = RDBI(ECUs[i],[k + j])
          if response is positive then
            ECUs[i].dids.append(k + j, len(response.value)
          end if
          j = j + 1
        end while
        ECUs[i].dids.append(k, len(response.value)
      else if response is responseTooLong then
        j ← 1
        while j ≤ n do
          response = RDBI(ECUs[i],[k + j])
          if response is positive then
            ECUs[i].dids.append(k + j, len(response.value)
          end if
          j = j + 1
        end while
      else if response is requestOutOfRange then
        pass
      end if
      k ← k + n
    end while
    i ← i + 1
  end while
```

2.3. Experiments for Data Collection

The results of the signal discovery pipeline provide a complete set of control devices and their reachable address space. To identify the best-performing data identifier for each targeted signal, multiple experiments are carried out.

The underlying concept of all experiments is to first perform two measurements at distinct observation points and filter out all data identifiers that remain constant for both experiments. This is carried out to significantly reduce the duration of each data collection, as most of the DIDs stay constant for all measurements within an experiment. For the remaining non-constant DIDs, additional measurements are performed to improve the availability of data for signal identification. We will refer to this strategy as the *pre-filtering strategy*.

Another important part of the data collection strategy is to collect data for the same ground truth at different times and avoid collecting consecutive measurements for the same condition. This is only necessary for categorical signals—i.e., brake pedal activation, chassis level, and selected gear, in general signals that have continuous physical representation—but can optionally be applied to any experiment. This is due to the fact that continuous signals, i.e., vehicle speed, steering wheel angle, accelerator pedal position, and charging power, are processed using regression learning, which is less vulnerable to a structured data collection approach. For categorical signals, machine learning is used during data identification, where this strategy is essential to avoid the model learning unrelated or misleading features caused by structured or time-correlated data collection [34].

For example, rather than collecting five measurements in gear Park (P) consecutively, one or two measurements are performed in P, followed by measurements in different gears, before returning to collect additional measurements in P. The reason for this strategy is that otherwise the Neural Network (NN) used (see Section 2.4.2) is potentially able to map an unrelated signal (e.g., DID representing state of charge (SOC)) to the ground truth. In other words, without the alternating pattern, unrelated DIDs might feature a structure, correlating to the different measurements without containing meaningful information for this task, e.g., gear signal identification.

As an example, the order of the measurements for the gear experiments (without distinguishing between the two different driving gears) is shown in Table 3. Overall, eight measurements per gear were collected. Instead of collecting eight consecutive measurements per DID for each gear, the measurements were divided into three separate groups (first one measurement, then three, and finally four). Diversifying the measurements makes it more challenging for an NN to learn unrelated signals (e.g., SOC), thereby facilitating the identification of correlated signals. If this strategy was not applied in this example, a DID containing the SOC would correlate to the selected gear without actual causation, e.g., high SOC for first measured gear to low SOC for last measured gear. This strategy was similarly applied in the data collection process of the other categorical signals, such as the brake pedal experiment. We will refer to this strategy as the *alternating measurements strategy*.

Table 3. Number of measurements per DID, distributed across gears and time steps.

Gear	Time Step											
	1	2	3	4	5	6	7	8	9	10	11	12
P	1						3					4
R		1				3			4			
N			1					3			4	
D				1	3					4		

2.3.1. Brake Pedal Experiment

The goal of the brake pedal experiment is to discover DIDs that indicate whether the brake pedal is activated or not (see [35]). The degree to which the brake pedal is pressed is not relevant. For this experiment, each DID is read ten times with the brake pedal not activated, and ten times with it activated. Since it is a categorical signal, the strategy of alternating measurements is applied. For both types of measurements, a person is seated in the stationary vehicle and either pressing or not pressing the brake pedal, ensuring similar conditions for both sets of measurements. Overall, the experiment consists of 20 measurements under two different conditions.

2.3.2. Chassis Level Experiment

Conveniently, the vehicle under study includes options in the control panel (Porsche Communication Management) for manually changing the ride height to four different levels: *Lift*, *Normal*, *Lowered*, and *Low*. To find a DID that corresponds well to the chassis level, each option is chosen, and eight measurements per DID are performed. As with the brake pedal experiment, the alternating measurements strategy is used. This results in a total of 32 measurements per signal. For this experiment, the vehicle remains stationary during all measurements.

2.3.3. Steering Wheel Angle Experiment

This experiment aims to identify DIDs corresponding to the steering wheel angle. While the vehicle remains stationary, measurements for seven different steering wheel angles are taken: 0°, ±90°, ±180°, and ±360°. The steering wheel is manually turned to the desired position before each measurement. For each steering wheel angle, five measurements are taken, resulting in a total of 35 measurements per DID.

2.3.4. Vehicle Speed Experiment

The vehicle speed experiment is one of the more time-consuming experiments. This makes signal filtering at an early stage all the more important. A single measurement of all available DIDs is performed at a constant speed of 10 km/h, and another at 100 km/h. Signals that remain constant across both conditions are filtered out to optimize the succeeding data collection.

A large gap between the two measured speeds was chosen to avoid mistakenly filtering out signals that change between the two driving gears of the vehicle under study. This also allows the collected data to be reused for other experiments, such as the gear experiment. Therefore, the selected gear throughout all the measurements in this experiment is Drive (D).

For the pre-filtered set of DIDs, data is collected for 12 different speeds from 10 km/h to 120 km/h, with 10 km/h increments. At each speed, five measurements per DID are taken, resulting in a total of 60 measurements per DID. Since the vehicle speed is a linear signal, the alternating measurements strategy is not applied.

During this and the following experiments, whenever the vehicle is not stationary, a *chassis dynamometer* (Renk GmbH, Augsburg, Germany) is used. While not essential for the experiments, it contributes to a repeatable, predictable, and safe environment. Another option would be to collect the data at different speeds on a highway.

2.3.5. Gear Experiment

The purpose of the gear experiment is to identify a DID that is able to differentiate not only between the different selectable gears, i.e., Park (P), Neutral (N), Reverse (R), and Drive (D), but also between the two different driving gears of the vehicle under study.

Eight measurements per DID were taken from the stationary vehicle for all four selectable gears. This collected data is used to identify a signal that differentiates well between the four selectable gears. As explained in the beginning of Section 2.3, the alternating measurements strategy is applied. The order of the performed measurements is shown in Table 3.

As described in Section 2.1, the VUS is equipped with a two-speed automatic transmission and thus two different driving gears. These driving gears are referred to as Drive 1 (D1) for low-to-moderate-speed conditions, and Drive 2 (D2) for moderate-to-high-speed operation. The driving gears cannot be selected manually but are instead shifted automatically by the VUS when operating in the selected drive mode D.

Therefore, another data collection is created by removing the measurements for the gear D from this data collection and replacing it with two different automatically crafted datasets for the gears D1 and D2. These two datasets are created using data from Section 2.3.4: For the D1 dataset, two measurements per speed are randomly taken for the speeds 10 km/h, 20 km/h, 30 km/h, and 40 km/h, resulting in eight measurements per DID for this gear. Accordingly, the D2 dataset is crafted for the speeds 90 km/h, 100 km/h, 110 km/h, and 120 km/h. In the reference study [27], it has been shown that for these speeds, the vehicle is in the desired driving gear.

This automatically crafted dataset is then used to identify DIDs that are also able to differentiate between the two driving gears. Additional measurements were taken to verify the identified DIDs in Section 3.3 by measuring data for negative speeds in gear R and both positive and negative speeds for gear N.

The resulting two datasets contain 32 values per DID for the four selectable gears, and 40 values per DID for the dataset with the two driving gears.

2.3.6. Accelerator Pedal Position Experiment

In contrast to the brake pedal experiment, the goal of the accelerator pedal position experiment is not only a binary detection, but the identification of a continuous signal (see [27]). The setup of this experiment is challenging, as it is difficult to provide a constant ground truth value manually.

One option would be the use of a pedal robot, providing constant pressure on the accelerator pedal. Even though a pedal robot was available, its use was deliberately avoided because the goal is a pipeline that is reproducible without requiring specialized equipment.

Therefore, throughout the data collection for this experiment, the vehicle is kept at a constant speed of 10 km/h, in gear D, with one person seated in the vehicle. Measurements for eleven different accelerator pedal positions are taken. The accelerator pedal position is maintained by applying a constant tractive force, which can be read from the chassis dynamometer. The chassis dynamometer is in speed mode to keep the vehicle speed constant. Measurements were performed for tractive force values ranging from 2 kN to 10 kN, with 1 kN increments. In addition, measurements were taken at the neutral position (0 kN) and with the accelerator pedal fully depressed. Measurements at 1 kN were skipped, as maintaining this position was deemed unfeasible. For each accelerator pedal position, two measurements were taken, resulting in 22 values per DID.

2.3.7. Charging Experiment

The objective of this experiment is to identify parameters associated with the performance and behavior of the battery system in general. The VUS has two charging ports placed on both sides forward of the front doors. The port located on the left side only supports alternating current (AC) charging, whereas the other port supports AC and direct current (DC) charging. In order to gather representative data for both charging methods,

two different experiments were conducted. For all charging experiments, the battery of the VUS must be sufficiently empty to support both the required charging power and time.

The AC charging experiments were conducted using a Pulsares SimpleBox (PULSARES GmbH, Nienstädt, Germany), a wall-mounted charging station with a so-called SMART control input, which allows the user to limit the maximum charging power. This input can be stimulated either via a pulse-width modulation (PWM) signal or by applying a constant voltage in the range of 0 and 3 V. This flexible input allows for precise adjustment of the charging power ([36] p. 26).

For the experiments, an ESP32-WROOM-32 (Espressif, Shanghai, China) was programmed to generate the required PWM signal. The underlying function given by the manufacturer for calculating the resulting amperage per phase is as follows [36]:

$$\text{Duty}[\%] = 3.077 \times \text{Amps} - 8.462 \quad (1)$$

The resulting characteristic curve is shown in Figure 3. In total, 5 AC charging experiments were conducted with the PWM duty cycle increasing by 10% starting at 11%, with 5 samples per experiment.

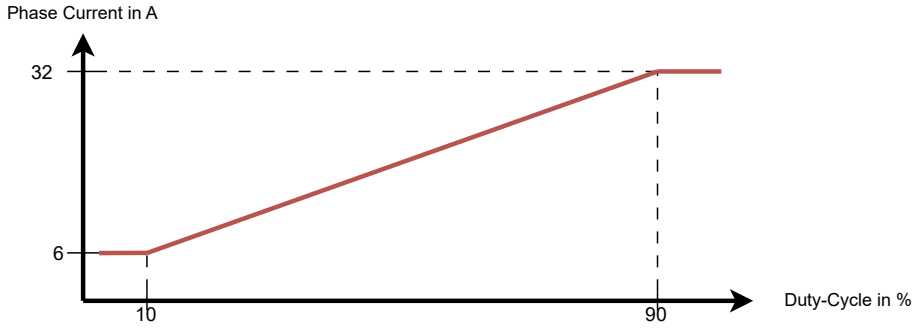

Figure 3. SMART PWM characteristic curve, with amperage per phase over duty cycle (Figure 13 in [36]).

The DC charging experiment was carried out using a 100 kW DC Charger (JEMA Energy SA, Lasarte-Oria, Spain). This charger offers an application programming interface (API) that logs detailed information about the charging process. Apart from intuitive values like charging voltage and power, the charger also exposes data that results from the communication between the charger and the car, most importantly, the SOC. As shown in (Figure 12 [27]), the charging power of VUS drops with increasing SOC, which automatically causes a change in the signals of interest, allowing for correlation with the signals provided by the charger.

2.4. Data Identification

The collection of data in different experiments described in Section 2.3 results in multiple datasets per experiment. Even with the pre-filtering strategy applied, the datasets still contain measurements for hundreds of different DIDs. Only a fraction of these DIDs contain valuable information related to the target signal. The goal of the data identification process is to detect the DIDs from which the desired information can be read best for each of the targeted signals.

Before the actual interpretation of raw measurements can begin, it is necessary to combine the individual measurements into a single file. During the data collection pipeline, for each distinct ground truth value, e.g., 10 km/h or 20 km/h, an experiment file is created. These experiment files contain the raw measurements for the specific ground truth value. Ground truth values are added as external measurements to each experiment

file, such that each measurement can be associated with the correct ground truth value. Then, all experiment files from a single experiment are concatenated into one combined file. These combined experiment files contain all the raw measurements, along with the associated ground truth values. With that first pre-processing step, the data identification process can begin.

2.4.1. Regression Learning

ISO 14229-1 defines a set of functions that vehicle manufacturers can use to encode the byte values behind data identifiers. Nine of the ten defined functions can be transformed into regular linear functions (C.6 in [26]). Furthermore, ref. [27] showed regular usage of linear functions for converting DID values to required measurement values. The challenge is to find a transfer function that optimizes the conversion from DID values to the measured ground truth values. Those DIDs, which, after conversion, show minimal deviation from the ground truth values, are most likely encoding the required measurement. To define a linear function, two unknowns have to be determined: slope and offset. By experiment design, more than two value and ground truth pairs exist for every given DID, over-determining the equation system. Furthermore, this also guarantees that only DIDs that correlate with the ground truth will show minimal error. Deriving the two unknowns can mathematically be expressed as a linear least-squares optimization problem as denoted in Equation (2) [37]. The dimensions of A and b directly follow from the number of samples available from the experiment.

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2, \quad \text{With} \quad A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ \vdots & \vdots \\ A_{n1} & A_{n2} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2}$$

Every line of A and b defines a linear equation in which $A_{n,1}$ holds the DID value and b holds the measured ground truth value. By definition $A_{n,2} = 1$. The values x_1 and x_2 are the slope and offset, which are being optimized. Up to this point, the DID values are byte arrays of arbitrary length. In order to transform them into useful numerical numbers, three important pieces of information are missing.

1. The byte order is unknown. For a given 16-bit value, it can be interpreted in either little-endian or big-endian format. Similarly, for 8-bit values, it is unknown if the most or least significant bit comes first.
2. The correct encoding of the values is unknown. A given 16-bit value could, for example, be an unsigned integer, a signed integer, or an IEEE 754 float.
3. The partition of the values is unknown. Manufacturers sometimes pack multiple values into a single DID value. A DID value with a total length of 48 bits could potentially encode four 8-bit values followed by a 16-bit value.

This makes it necessary to calculate the error for all permutations of encodings, byte orders, and partitions of all DID values. In the analysis, the following encodings are considered in both byte/bit orders:

1. 8-bit: u_int 8, int 8.
2. 16-bit: u_int 16, int 16, float 16.
3. 32-bit: u_int 32, int 32, float 32.
4. 64-bit: u_int 64, int 64, float 64.

2.4.2. Machine Learning

While regression learning works well for identifying continuous signals, it has major limitations when applied to categorical data. The problem is that categories might not have a natural numeric relationship. For example, for the gear experiment, there is probably no mapping from the different gears to natural numbers that correctly represents the relationship. For other categorical signals, mapping the categories to numerical values might be possible (e.g., chassis level), since the categories are based on a continuous variable, such as the ride height. Regression learning is not able to split data into discrete classes, and its output might predict invalid values, e.g., a predicted gear value of 2.5. Therefore, a second data identification strategy based on machine learning was introduced to address these limitations.

Machine learning offers several benefits for categorical signal identification. To name a few, machine learning models treat categories as distinct classes and do not make assumptions about numeric relationships. They also support multi-class classification. Last but not least, they can learn complex, non-linear patterns in high-dimensional feature spaces.

The goal is to identify DIDs that encode categorical signals, i.e., brake pedal activation, chassis level, and gear selection. To achieve this, the first step is to load the data.

The data is loaded from the previously created combined experiment file for a specific experiment (e.g., gear experiment) into a *Pandas DataFrame*. The columns of the DataFrame are 'Server ID', 'DID', and 'Ground Truth Value', followed by the data columns. While the columns 'Server ID' and 'DID' identify the DID, the column 'Ground Truth Value' contains the categorical ground truth value, for the gears, e.g., 'P' and for brake pedal activation, 'True' or 'False'). The number of data columns depends on the number of measurements per ground truth value. For example, in the gear experiment, eight measurements per DID were performed, and thus the DataFrame includes eight value columns.

Thus, the number of columns n_{columns} and the number of rows n_{rows} in the DataFrame can be expressed as

$$n_{\text{columns}} = 3 + n_{\text{samples}} \quad \text{and} \quad n_{\text{rows}} = n_{\text{DIDs}} \times n_{\text{classes}}, \tag{3}$$

where n_{samples} is the number of measurement values per class and per DID, n_{DIDs} is the number of DIDs, and n_{classes} is the number of ground truth classes.

After loading, the data is split into a training set and a validation set. During data collection, the distribution of measurements is controlled to ensure an equal number of data points for each ground truth value. To maintain this balance, the data is split row-wise. First, the data in each row is shuffled. Next, at least 75% of the data is added to the training set. The remaining data is added to the validation set. As a result, the same number of data points per ground truth value and per DID is available in both sets. Additionally, the structure of the data is slightly modified during the split. Each row now corresponds to a measurement value. The columns are 'Server ID', 'DID', 'Ground Truth Value', and a single 'Value' column.

Following the splitting of the data, further pre-processing steps are needed for both the training and the validation sets. To avoid redundancy, pre-processing is explained only for the training set, but the exact same process is performed for the validation set. The DataFrame containing the training set is split by DID into multiple DataFrames. These DataFrames are stored in a dictionary, indexed by a tuple of 'Server ID' and 'DID'. At this point, the 'Value' column still contains raw data in the form of byte lists. This value column is expanded into multiple value columns. The number of value columns after expansion depends on the length of the DID and is determined individually for each DID. The last pre-processing step is *One-Hot Encoding*. This technique transforms the 'Ground Truth Value' column into multiple binary columns, one for each class. This converts the

categorical labels into a binary indicator matrix, where each class has its own column [38]. The results of the loading and pre-processing steps are two dictionaries of DataFrames ready for the training process.

The data is split into individual DataFrames per DID to enable independent training for each. The overall goal is to identify a single DID per experiment that contains the complete information about the signal, not a combination of multiple DIDs. This is achieved by iterating through all the DIDs of an experiment and training a separate instance of the same NN architecture for each one. To reduce training time and to avoid flawed results, a final filtering step is applied. All DIDs that are ambiguous—i.e., they have overlapping values between classes—are removed. For example, if a DID has the same value in 'N' and in 'P', it is not a suitable candidate. The training process uses a fully connected NN consisting of one hidden layer with 16 neurons. The number of neurons on the input layer depends on the length of the individual DID, i.e., one neuron per byte. For the output layer, the number of neurons corresponds to the number of classes of the experiment, e.g., two for the brake pedal activation experiment. Each DID is trained individually on this shared NN architecture with a batch size of 64 for 100 epochs. After training all DIDs (without the previously removed ones) and storing each model's metrics in a dictionary, the evaluation process can begin.

The performance metrics used for evaluation are *Accuracy*, *Precision*, *Recall*, and *F1-Score*. These are standard machine learning metrics used to evaluate classification models. All of them are derived from the confusion matrix (see Table 4), which is particularly useful in binary classification tasks. In multi-class classification, the confusion matrix remains useful when evaluated on a per-class basis.

Table 4. Confusion matrix for classification tasks in machine learning.

Actual	**Predicted**	
	Positive	**Negative**
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negativee (TN)

For example, for the gear experiment and class gear P, the confusion matrix values are defined as follows:

- **TP:** Predicted P when true class is P.
- **FP:** Predicted P but true class is N, R, or D.
- **FN:** Predicted N, R, or D but true class is P.
- **TN:** Predicted N, R, or D and true class is N, R, or D.

The metrics Accuracy, Precision, Recall, and F1-Score are calculated by these values:

$$f_{\text{accuracy}} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4}$$

$$f_{\text{precision}} = \frac{TP}{TP + FP}, \tag{5}$$

$$f_{\text{recall}} = \frac{TP}{TP + FN}, \tag{6}$$

$$f_{\text{F1}} = \frac{2 \times f_{\text{precision}} \times f_{\text{recall}}}{f_{\text{precision}} + f_{\text{recall}}}. \tag{7}$$

In summary, Accuracy is the rate of total correct predictions out of all predictions made, Precision is the rate of correctly predicting that class out of all predictions made for

that class, Recall is the rate of correct class predictions out of all samples of this class, and F1-Score combines Precision and Recall into a single metric [39].

Since each of these metrics captures a different aspect of classification performance, they are combined into a single weighted score, enabling consistent ranking across all DIDs:

$$f_{\text{Score}} = 0.4 \times f_{\text{Accuracy}} + 0.2 \times f_{\text{Precision}} + 0.2 \times f_{\text{Recall}} + 0.2 \times f_{\text{F1}}. \quad (8)$$

Accuracy is weighted twice as heavily as the other metrics, which are weighted equally, since the classes are balanced and correctness is the most important factor. Also, it overcomes the issue of Precision and Recall, being already included in the F1-Score, which would otherwise be effectively double-counted. The DIDs are sorted by their performance based on this score function. If a clear top candidate is identified, the objective of the pipeline is considered fulfilled. Otherwise, the best scoring DIDs are kept for further data collection. A threshold of 0.7 is defined as the minimum score for selection, as it balances model confidence and flexibility. This ensures that only high-performing DIDs are retained, resulting in a reduced duration of additional data collection, avoiding the exclusion of potentially valuable candidates. For the selected top candidates, additional data collection experiments are performed, and the new data is fed back into the individually trained instances of the same NN architecture to enhance training and identify a single DID.

Although it is possible to apply the machine learning pipeline to continuous signals, it has significant limitations. Continuous signals often contain fluctuations and sensor noise. Machine learning models tend to overfit to such noisy patterns, especially given that the datasets are very small from a machine learning perspective. In addition, machine learning might identify DIDs that are merely correlated with the target signal, rather than the true source. Since regression learning provides a more direct interpretation of continuous signals, the machine learning approach was not applied to continuous signals.

3. Results, Evaluation, and Discussion

After describing the VUS, the hardware and software setup, and the experimental methods, we now turn to the results of the proposed pipelines. We begin with the results of the vehicle discovery pipeline and continue with the data identification pipelines. As explained in Section 2.4, this process is split into two parts. We first present the results for continuous linear signals and conclude with those for categorical signals.

3.1. Results of Discovery

The overall goal of the discovery pipeline is to identify all reachable ECUs, services, and DIDs within the vehicle. The results of this pipeline are a prerequisite for signal identification.

3.1.1. Connection and Communication Establishment

The initial DoIP connection via OBD-II and Ethernet proved to be 100% reliable. Since the OBD-II connector must be located under the driver's side dashboard, it is easily accessible. As a result, physically connecting the vehicle to the host computer takes only a few seconds.

For the initial connection, the vehicle announcement message is necessary. There were no cases in which the vehicle failed to send this message. It can take a few seconds until the message is sent by the vehicle and received by the laptop, but the delay never exceeded ten seconds.

The vehicle may change the Gateway ECU's IP address from day to day, making re-initialization necessary. However, due to the fast connection setup, this is not a significant issue. This behavior can also be prevented by routing the connection through a Dynamic

Host Configuration Protocol (DHCP) router. The router can assign and persist logical addresses over multiple days, even if both the vehicle and the laptop are disconnected.

3.1.2. Logical Address Discovery Results

The logical addresses are necessary for the server discovery. The vehicle announcement message already provides information about the logical address of the Gateway ECU and, as described earlier, the IP address of the Gateway ECU. The actual discovery process here is limited to discovering usable addresses for the laptop.

For the VUS, we were able to discover five different addresses for potential use by the laptop. The results are shown in Table 5.

Table 5. Discovered arbitration ID pairs.

Target Logical Address	Source Logical Address
0x4010	0x0E80
0x4010	0x0E81
0x4010	0x0E82
0x4010	0x0E83
0x4010	0x0E84

In this table, the target logical address corresponds to the Gateway ECU, while the source logical addresses represent the different values that can be assigned to the laptop during communication. Since the addresses range from 0x0E80 to 0x0EFF, which is relatively small, the discovery completes in approximately 2 min.

The logical address discovery proved to be stable and reliable. Multiple runs from different laptops resulted in the same addresses. Only the Gateway ECU's IP address will change for different laptops. Therefore, it is stored separately from the remaining vehicle data in a configuration file.

Discovery time could be reduced by stopping after the first valid laptop address is found, since for the server discovery, only one pair of identifiers (IDs) is needed. In our case, we would find a candidate with the test of the first address (0x0E80). However, due to the small duration compared to the overall pipeline, this optimization was omitted to ensure completeness.

3.1.3. Server Discovery Results

After retrieving the logical addresses of the Gateway ECU and establishing communication, the goal of the server discovery is to identify the addresses of all other reachable ECUs. These ECUs are identified as servers within the BEV's communication network. As described in Section 2.2.3, this is achieved by scanning the logical address space and considering successful session establishments as reachable ECUs.

In total, the server discovery (identifying reachable ECUs) for the VUS took 9 h 10 min. This includes the discovery of the complete set of reachable ECUs by probing each possible logical address. This approach guarantees the completeness of the server discovery process.

For the VUS, 41 ECUs were discovered. Table A1 shows the complete list of discovered ECUs for the VUS. The contained information about spare part number and software version for each ECU is retrieved by requesting the specific DIDs as defined in ISO 14229-1 (Table C.1 in [26]) (*vehicleManufacturerSparePartNumberDataIdentifier* (0xF187) and *vehicleManufacturerECUSoftwareVersionNumberDataIdentifier* (0xF189)).

Multiple runs of the server discovery consistently identified all 41 ECUs. The discovery time across different runs consistently ranged between 9 h 05 min and 9 h 15 min for the VUS.

The presented approach is brute-forcing the entire address space (0x0000 to 0xFFFF, 65,536 potential addresses), thus leaving room for optimization. ISO 13400-2 specifies the range and meaning of different logical address values used in DoIP (Table 13 in [33]). This restricts the possible address space for Vehicle Manufacturer (VM)-specific servers (ECUs in the vehicle) to the following ranges:

- 0x0001 to 0x0DFF (3583 addresses).
- 0x1000 to 0x7FFF (28,672 addresses).

This would reduce the number of potential addresses to 32,255. In fact, the discovered ECUs addresses for the VUS are all located in these defined sections. Limiting the discovery to the ISO-defined VM-reserved address space would halve the runtime to approximately 4 h 35 min. To ensure maximum completeness, we chose not to restrict the address space and instead performed the discovery across the entire logical address range.

3.1.4. DID Discovery Results

The previously discovered list of reachable ECUs enables the actual discovery of used DIDs, needed for data identification. For each ECU, the *ReadDataByIdentifier* UDS service is used to sequentially request all possible DID values and store the ones with a positive response (see Section 2.2.4).

The results of the DID discovery for the VUS are displayed in Table A1. The number of reachable DIDs varies significantly between different ECUs, ranging from low double-digit numbers to medium three-digit numbers. The discovery time is also subject to strong fluctuations, varying from several minutes to several hours. In total, 6280 reachable DIDs were discovered, which took approximately 19 h (excluding the time for the server discovery).

In Section 2.2.4, we also introduced a second algorithm for DID discovery, enabling parallel requesting of DIDs (see Algorithm 2). The algorithm supports requesting an in theory unlimited number of DIDs from the same ECU in parallel, but for the VUS, the number of DIDs is limited to eight, as allowed by ISO 14229-1 (Section 11.2 in [26]). We performed complete DID discovery runs using batches of eight (*8-Batch*) and four (*4-Batch*), meaning eight or four DIDs were requested in parallel.

Algorithm 2 was introduced to reduce the overall runtime of the DID discovery. For the 8-Batch, the runtime was reduced to 5 h 8 min. However, it was not able to discover the complete set of DIDs per ECU. While working correctly for most ECUs, it was only able to discover 6138 DIDs in total, missing 142 DIDs. For example, the number of discovered DIDs for ECU 0x4010 was only 353 instead of the expected 463. This behavior could not be prevented for the affected ECUs, resulting in dismissal of the 8-Batch. However, the 4-Batch was able to discover the complete set of DIDs, while reducing the runtime to 7 h 18 min, reducing the discovery time by 61.8%. Therefore, the use of the parallel method using batches of four is recommended, balancing speed and quality of outcome. The detailed performance of the parallel 4-Batch method per ECU is also shown in Table A1.

For the ECU with the logical address 0x4014, we were unable to complete the DID discovery. Even with application of the 8-Batch, the complete run time of the DID discovery for that ECU was estimated to take 211 h and thus deemed unfeasible. This occurs because the ECU is not required to respond, causing each request to end in a timeout. Despite the missing discovery for that server, we were not limited, as the required signals were also available through other ECUs, demonstrating functional redundancy within the system.

After presenting the discovery results, two optimization strategies are considered. With the parallel discovery algorithm, we have already presented a successful optimization. An additional opportunity for improvement is the order of the pipelines. It is not necessary to complete the server discovery before starting the DID discovery. The DID discovery process for an ECU can start as soon as its logical address is discovered. Therefore,

the server discovery could be interrupted after each newly discovered ECU by directly discovering all DIDs on this ECU. This enables the opportunity to progress to the data collection part with incomplete, but maybe sufficiently large, sets of ECUs and DIDs at the expense of completeness.

Another proposal for optimization is to limit the probed DIDs for each ECUs to the ranges defined in ISO 14229-1 (Table C.1 in [26]). Depending on the performed experiment, only specific ranges could be probed. For example, the ranges of VM-specific DIDs that were used are defined as follows:

- 0x0100 to 0xA5FF (42,240 parameters).
- 0xA800 to 0xACFF (1280 parameters).
- 0xB000 to 0xB1FF (512 parameters).
- 0xC000 to 0xC2FF (768 parameters).
- 0xCF00 to 0xEFFF (8448 parameters).
- 0xF010 to 0xF0FF (240 parameters).

In total, there are 53,488 parameters in these ranges. By reducing the number of parameters to probe by 12,048, an expected performance gain of approximately 18.4% could be achieved. Although this is difficult to estimate, since the DID discovery pace is not constant during the discovery in one ECU. Prioritizing completeness over efficiency, we decided against limiting the probed parameters.

3.2. Results of Linear Signals

As explained in Section 2.4.1, the continuous signals can be analyzed based on regression learning. The following sections present and analyze the results of the linear signal experiments.

3.2.1. Steering Wheel Angle Experiment Results

For this experiment, the pre-filtering strategy was not applied, resulting in a runtime of 5 h 41 min. Of the 6263 known DIDs, only 700 remained non-constant. With the pre-filtering strategy, a theoretical time saving of approximately 62% could have been achieved. Figure 4 shows this experiment's results. The invisible lines (ground truth and best match) do exactly match the original signal.

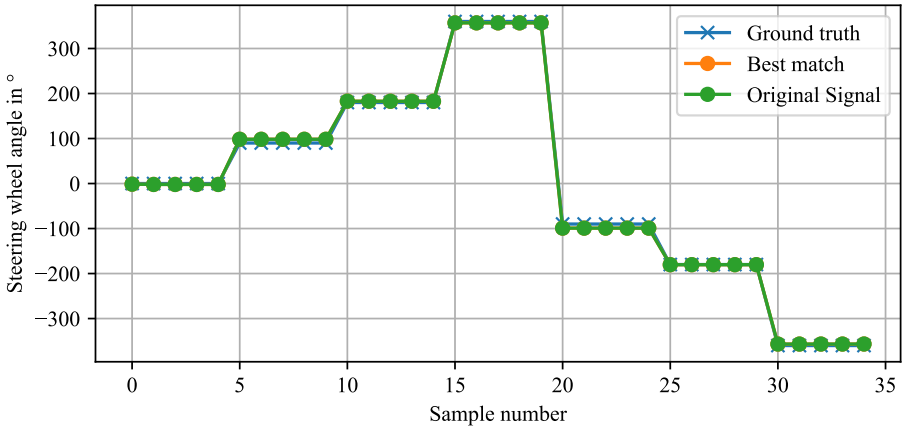

Figure 4. Discovered steering wheel angle signals and ground truth signal.

In total, two DIDs showed very good performance and therefore very few errors. The best signal DID 0x2B29 is located on the ABS Pump (0x4013); the other signal DID 0x1F0F is located on the Power Steering Module (0x4012) and is the original signal that was used

in [27]. The two discovered signals in the figure perfectly match the ground truth despite the fact that the steering wheel was manually placed into the intended positions.

3.2.2. Vehicle Speed Experiment Results

Of the known DIDs, only 988 were non-constant over the course of all measurements. In total, 20 signals have been found that performed equal or better than the signal identified in [27]. This original signal is located on Body Control (0x408B), which we were able to rediscover under its DID, 0x100E. The best signal DID, 0x22D2, is located on the Rear Right Radar Sensor (0x404E). As shown in Figure 5, the overall difference between the two signals is relatively small, and both match the ground truth very well.

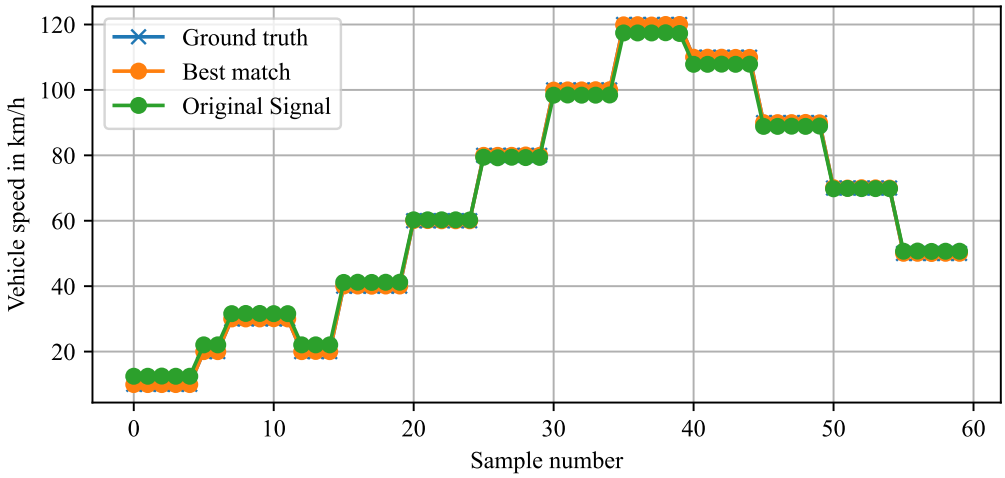

Figure 5. Discovered vehicle speed signal and ground truth signal.

3.2.3. Accelerator Pedal Position Experiment Results

Of the known DIDs, only 678 were non-constant over the course of all measurements. In total, four signals with acceptable performance for the accelerator pedal position were found. Figure 6 shows the second-best signal, which was used in [27].

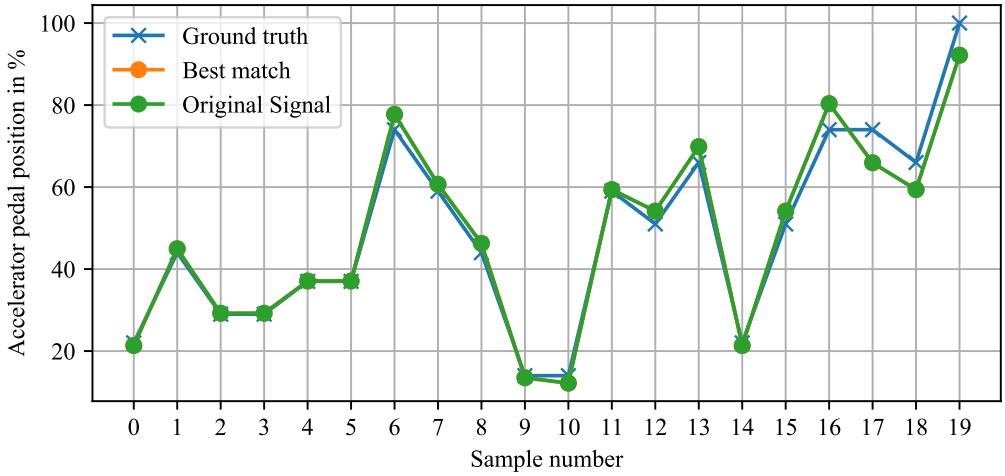

Figure 6. Discovered accelerator pedal position signal and ground truth signal.

This signal is located on ABS control (0x4013) and we were able to rediscover it as well under its original DID, 0x2B2F. The best signal DID, 0xF49A, is located on Thermal Management (0x4042). The overall difference between the two signals is not visible within the graph. The deviations from the ground truth are caused by two factors. Firstly, the

driver has to manually control the pedal, which introduces some noise (Samples 16 and 17). Secondly, the VUS was not able to hold the maximum force for the duration needed for two measurements (Sample 19). Nevertheless, the approach was able to successfully determine correct signals.

3.2.4. Charging Experiment Results

For the AC charging measurement suite, a total of 889 signals had to be considered. For the AC charging power, a perfectly matching signal was found on the charger of the High-Voltage Battery (0x4040) under DID 0x1DD7, as can be seen in Figure 7.

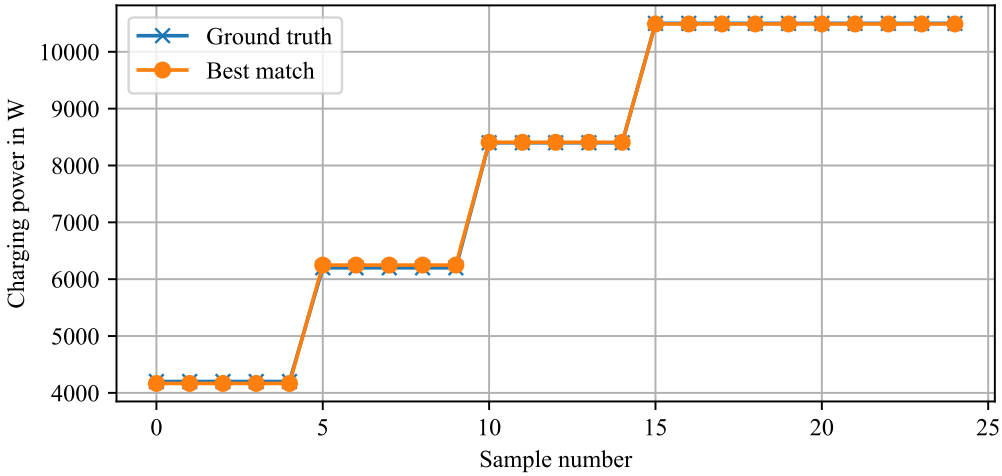

Figure 7. Discovered AC charging power signal and ground truth signal.

For the SOC, in total, 104 signals that performed equal or better than the original signal from [27] have been discovered. The DID was reconfirmed. The majority of good-performing signals are either located on the Battery Control (0x407B) or on the Charger of the High-Voltage Battery (0x4044). The overall difference between these signals is negligible.

3.3. Results of Categorical Signals

For the categorical signals, a different data identification strategy was applied due to their discrete nature. The neural network architecture and identification process for these signals were described in Section 2.4.2. Section 2.3 outlined the experimental setup for these signals, including the application of the pre-filtering and alternating measurements strategies. The categorical signals investigated in this paper are brake pedal activation, chassis level, and gear selection. The following subsections present the results for each signal individually.

3.3.1. Brake Pedal Activation Results

The application of the pre-filtering strategy limits the investigated DIDs to 487, reducing the runtime of the data collection process by 92.2%. The combined training time for these DIDs is only 17 s. For 46 DIDs, the classification results are perfect with 100% Accuracy, Precision, Recall, and F1-Score. This is expected, since classification of binary signals, like brake pedal activation, is usually less challenging.

The high-performing DIDs are concentrated on Brake Booster (0x403B, 5 DIDs) and Body Control (0x408B, 7 DIDs), but are also located on other ECUs. To validate the results, two of the DIDs with perfect scores were selected for further investigation. The selection criteria for the representative DIDs were a short response length and the semantic relevance

of the ECU. While some DIDs have a response length of up to thousands of bytes, others consist of a single byte. The first one is DID 0x0286 on Brake Booster (0x403B) with a response length of one byte. A byte value of 134 indicates activation, and a value of 135 indicates deactivation. The second one is DID 0x2B3A on ABS Pump (0x4013), where the single byte returns only the values 0 and 1. While 0 indicates that the brake pedal is activated, 1 indicates deactivation.

Attempted comparisons with known DIDs from the reference study [27] were unsuccessful, since the used DID 0xFD11 on the Brake Booster is not a DID identified in the vehicle discovery pipeline. This could be due to access restrictions requiring a different UDS session (e.g., extended session) or a mapping discrepancy between CAN and DoIP.

Nevertheless, multiple valid DIDs were found, supporting a robust identification process for the brake pedal activation signal. Not all DIDs were validated, but many likely represent the same ground truth. Due to the simplicity of the signal, high classification performance was expected.

3.3.2. Chassis Level Results

For the chassis level signal, 537 DIDs were investigated following the application of the pre-filtering strategy, resulting in a 91.4% faster data collection process. Training for these DIDs completes within 45 s. The classification results are perfect for two of the DIDs: DID 0x4506 and 0x454A on the Chassis Level Control Unit/Suspension (0x4080).

Table 6 displays the full list of measurement and ground truth values for DID 0x454A with a byte length of five. Byte zero directly encodes the chassis level state: 106 for Lift, 101 for Normal, 98 to 99 for Lowered, and 96 to 97 for Low.

Table 6. Chassis level states and Byte_0 to Byte_4 values for DID 0x2B94 from Chassis Level/Suspension Control (0x4080).

#	GT_Lift	GT_Normal	GT_Lowered	GT_Low	Byte_0	Byte_1	Byte_2	Byte_3	Byte_4
1	True	False	False	False	106	219	54	201	175
2	True	False	False	False	106	155	54	205	175
3	True	False	False	False	106	155	54	205	175
4	True	False	False	False	106	155	54	205	175
5	True	False	False	False	106	219	54	201	175
6	True	False	False	False	106	155	54	205	175
7	True	False	False	False	106	155	54	205	175
8	True	False	False	False	106	219	54	201	175
9	False	True	False	False	101	217	230	137	157
10	False	True	False	False	101	218	6	141	158
11	False	True	False	False	101	217	230	137	157
12	False	True	False	False	101	217	230	137	154
13	False	True	False	False	101	217	230	137	154
14	False	True	False	False	101	217	230	137	155
15	False	True	False	False	101	217	230	137	155
16	False	True	False	False	101	217	230	137	157
17	False	False	True	False	98	217	102	89	149
18	False	False	True	False	99	89	86	97	147
19	False	False	True	False	98	217	102	89	149
20	False	False	True	False	98	217	102	89	149
21	False	False	True	False	98	217	102	89	149
22	False	False	True	False	99	89	86	97	147

Table 6. *Cont.*

#	GT_Lift	GT_Normal	GT_Lowered	GT_Low	Byte_0	Byte_1	Byte_2	Byte_3	Byte_4
23	False	False	True	False	98	217	102	89	149
24	False	False	True	False	99	89	86	97	147
25	False	False	False	True	96	216	150	53	136
26	False	False	False	True	96	216	150	53	136
27	False	False	False	True	97	152	230	49	138
28	False	False	False	True	97	152	230	49	138
29	False	False	False	True	96	216	150	53	136
30	False	False	False	True	96	216	150	53	136
31	False	False	False	True	96	216	150	53	136
32	False	False	False	True	97	152	230	49	138

Overall, there are eleven DIDs with strong classification results, each achieving a score of above 70%. Four of them are located on the Chassis Level Control Unit/Suspension (0x4080, 4 DIDs), including the previously identified DID 0x2B94 from the reference study [27] with a byte length of one and a score of 70.8%. The mapping of chassis level states and byte zero values is shown in Table 7.

Table 7. Chassis level states and Byte_0 values for DID 0x2B94 from Chassis Level Control Unit/Suspension (0x4080).

#	GT_Lift	GT_Normal	GT_Lowered	GT_Low	Byte_0
1	True	False	False	False	80
2	False	True	False	False	64
3	False	False	True	False	48
4	False	False	False	True	32

3.3.3. Gear Selection Results

The application of the pre-filtering strategy during the gear selection experiment reduced the number of investigated DIDs to 646. This reduced the data collection runtime by 89.6%.

For distinguishing between the four selectable gears, four DIDs with a perfect score were identified. Three of these DIDs are located on the vehicle control unit, while the other one is on the body control module. Further investigation of these signals confirms the correctness of the results: The byte-to-gear mapping for three of them is as follows: P = 5, R = 6, N = 7, and D = 8. For the last one on the vehicle control unit, the mapping follows a reverse order: P = 8, R = 7, N = 6, and D = 5.

For additional distinction between the two different driving gears D1 and D2, an additional dataset was created as described in Section 2.3.5. This time, one DID with a perfect score was detected, located on the gateway ECU, along with four other signals with a score of above 90%. The identified byte four mapping for the four selectable gears in the DID with a perfect score is the following: P = 81, R = 96, N = 112, and D = 128. The differentiation between D1 and D2 was accomplished by comparing the values in bytes two and three, but no consistent pattern was identified. Since manual validation was unsuccessful, validating the results would require additional data.

The reference study [27] did not identify a single signal containing this information. Instead, two separate signals were used.

4. Summary and Conclusions

In this study, we present a holistic and practical approach to accessing internal vehicle communication via the Automotive Ethernet bus system and UDS protocol. This approach is automated until and including signal identification, with the user ultimately selecting the signals for further analyses. We applied this methodology to signals of interest according to [35] with respective experiment design and ground truth data capturing for signal alignment. The identified signals were evaluated against the applied signals in [27]. This methodology presents a modular design that can be extended to further signals. With our tool, a cost-efficient approach is presented that only requires standardized equipment as described in this study. The major discoveries of this study can be summarized as follows:

- **Discovery of potential addresses for signal identification.**
  A total of 41 potential ECU addresses with reference to their spare part number and function were identified, and subsequently, 6280 responsive DID addresses were identified and analyzed. This process has been performed multiple times to guarantee reproducibility, which was successful for the 4-Batch approach.
- **Regression learning for linear signals.**
  The continuous signals representing physical values were identified with a linear regression-based learning approach, applying a slope and an offset to the signals to mirror the ground truth data captured during the experiments.
- **Machine learning for categorical signals.**
  Since categorical signals cannot be decoded linearly, we applied a machine learning approach to identify the respective categories for the signals of interest.
- **Optimization strategies for reduced computational effort.**
  The primary process of our study is time-consuming. Thus, optimization strategies were applied to reduce computational time. The respective optimization techniques were described in the study, from the parallel discovery algorithm (see Algorithm 2) to pre-filtering constant signals in the capturing process to collecting multiple targeted signals in single experiments and ultimately restricting the process to servers and DID ranges specified in the norm. These helped reduce computational effort and thus testing time significantly.

Author Contributions: N.R.: Conceptualization, methodology, investigation, resources, writing—original draft, visualization, writing—review and editing, project administration. N.R. is the first author. N.H.: Investigation, writing—original draft, writing—review and editing, visualization. A.M.: Investigation, writing—original draft, writing—review and editing, visualization. M.L.: Resources, supervision, writing—review and editing, funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within the project "ScaleUp-eDrive" under grant number 16THB0006C.

Data Availability Statement: We want to encourage researchers and engineers from the automotive industry to understand our work and to enable them to apply our methodology to their test vehicles, identify signals of interest, and conduct analyses on the vehicle level. Therefore, our methodology is provided as open-source and accessible via FTMGithub, accessed on 23 August 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Discovery Results

The results of the ECU and the DID discovery are shown in Table A1. Note that some ECUs are reachable via different logical addresses with independent sets of DIDs.

Table A1. Overview of all discovered ECUs with hardware and software configuration and DIDs with discovery times for sequential and parallel (4-Batch) discovery per ECU.

#	ECU Address	Name	Spare Part Number (Software Version)	DIDs	DID Discovery Time in hh:mm	
					Sequential	Parallel (4-Batch)
1	0x400b	Tire Pressure	992907273D (0220)	78	0:11	0:03
2	0x400c	Steering Column	9J1953502M (0008)	26	0:22	0:11
3	0x400e	Body Control	971907063P (0728)	349	0:25	0:12
4	0x4010	Network Gateway	9J1907468K (0708)	463	0:05	0:04
5	0x4012	Power Steering	9J1907445K (0450)	70	0:26	0:11
6	0x4013	ABS Pump	9J1614095S (0190)	178	0:27	0:19
7	0x4014	Instrumentation | Dashboard	9J1920901AJ (0667)	-	-	-
8	0x4015	Safety | Airbag	992959655D (3303)	419	0:23	0:12
9	0x401c	Internal Engine Sound	9J1035446H (0301)	53	0:22	0:08
10	0x4023	Tailgate	8W2959107A (0335)	94	0:11	0:03
11	0x4024	Electric Spoiler	992907483T (0170)	61	0:12	0:03
12	0x403b	Brake Booster	9J1907107G (0190)	60	0:32	0:11
13	0x403e	Rear Driver Door	4M0959795N (0390)	60	3:18	1:15
14	0x403f	Rear Passenger Door	4M0959795N (0390)	61	3:18	1:14
15	0x4042	Thermal Management	4KE965429M (0325)	280	0:12	0:04
16	0x4044	HV Battery Charger	9J1915681BT (1272)	227	0:11	0:03
17	0x4046	Air Conditioning	9J1907040L (1410)	154	0:11	0:03
18	0x404a	Front Driver Door	4M0959793N (0390)	87	0:11	0:03
19	0x404b	Front Passenger Door	4M0959792N (0390)	73	0:11	0:03
20	0x404c	Seat Control	4M6959760 (0064)	105	0:11	0:11
21	0x404e	Rear Right Radar	4N0907566AM (0588)	100	0:11	0:03
22	0x404f	Driver Assist	4K4907117H (0371)	277	0:27	0:12
23	0x4053	Gear Selector	9J1713033E (0300)	35	0:11	0:03
24	0x4064	External Engine Sound	9J1035335J (0111)	53	0:22	0:08
25	0x4067	Emergency Call	4N0035282C (0450)	151	0:17	0:04
26	0x4073	Multimedia System	9J1035070BF (3882)	195	0:21	0:05
27	0x4076	Vehicle Control	9J1909101DG (0021)	389	0:26	0:12
28	0x407b	Battery Control	9J1915234AS (1646)	301	0:23	0:06
29	0x407c	Electric Drive Motor	9J1907121BN (0023)	67	0:32	0:08
30	0x4080	Chassis level	9J1907553R (1510)	146	0:28	0:12
31	0x4086	Online Services	9J1907018AL (1810)	170	0:13	0:04
32	0x408a	Rear Left Radar	4N0907566AM (0588)	77	0:11	0:03
33	0x408b	Body Control	992907064DK (0604)	473	0:15	0:11
34	0x4096	Left LED Headlight	992941572BA (9002)	69	0:11	0:06
35	0x4097	Right LED Headlight	992941572BA (9002)	69	0:11	0:06
36	0x40a5	Coupling Antenna	9J1035504B (0002)	22	0:25	0:06
37	0x40b7	DC-DC Converter	9J1959663BG (1910)	102	0:55	0:14
38	0x40c7	HV Booster	9J1915539DE (1910)	132	0:55	0:14
39	0x40f1	Safety | Airbag	992959655D (3303)	419	0:24	0:12
40	0x4767	Left LED Headlight	992941572BA (9002)	66	0:12	0:08
41	0x4768	Right LED Headlight	992941572BA (9002)	66	0:12	0:08
Total				**6280**	**19:04**	**7:17**

References

1. International Energy Agency. Global EV Outlook 2024. Available online: https://www.iea.org/reports/global-ev-outlook-2024 (accessed on 23 August 2024).
2. European Union Council. Regulation (EU) 2021/1119 of the European Parliament and of the Council of 30 June 2021 Establishing the Framework for Achieving Climate Neutrality and Amending REGULATIONS (EC) No 401/2009 and (EU) 2018/1999 ('European Climate Law'). Available online: http://data.europa.eu/eli/reg/2021/1119/oj (accessed on 23 August 2024).
3. Ministry of Ecology and Environment of the People's Republic of China. China's Policies and Actions for Addressing Climate Change. Available online: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://english.mee.gov.cn/Resources/Reports/reports/202211/P020221110605466439270.pdf&ved=2ahUKEwj1k_y21YqIAxVR3QIHHTsFAa8QFnoECBgQAQ&usg=AOvVaw3WeiSi_sDkHB7h_55p_I56 (accessed on 23 August 2024).
4. Environmental Protection Agency. Multi-Pollutant Emissions Standards for Model Years 2027 and Later LightDuty and Medium-Duty Vehicles. Available online: https://www.govinfo.gov/content/pkg/FR-2024-04-18/pdf/2024-06214.pdf (accessed on 23 August 2024).

5. Rapson, D.; Muehlegger, E. The Economics of Electric Vehicles. *Rev. Environ. Econ. Policy* **2023**, *17*, 274–294. [CrossRef]
6. Lienert, P.R. Exclusive: Automakers to Double Spending on EVs, Batteries to $1.2 Trillion by 2030. 2022. Available online: https://www.reuters.com/technology/exclusive-automakers-double-spending-evs-batteries-12-trillion-by-2030-2022-10-21/ (accessed on 27 March 2025).
7. European Alternative Fuels Observatory (EAFO). Electric Vehicle Model Statistics, 2024. Available online: https://alternative-fuels-observatory.ec.europa.eu/policymakers-and-public-authorities/electric-vehicle-model-statistics (accessed on 22 April 2025).
8. Rosenberger, N.; Fundel, M.; Bogdan, S.; Köning, L.; Kragt, P.; Kühberger, M.; Lienkamp, M. Scientific benchmarking: Engineering quality evaluation of electric vehicle concepts. *e-Prime-Adv. Electr. Eng. Electron. Energy* **2024**, *9*, 100746. [CrossRef]
9. Zairi, M. *Benchmarking for Best Practice: Continuous Learning Through Sustainable Innovation*; Routledge: London, UK, 1998.
10. Camp, R. *Benchmarking: The Search for Industry Best Practices that Lead to Superior Performance*; Productivity Press: New York, NY, USA, 2006.
11. Bhadani, R.; Bunting, M.; Nice, M.; Tran, N.M.; Elmadani, S.; Work, D.; Sprinkle, J. Strym: A Python Package for Real-time CAN Data Logging, Analysis and Visualization to Work with USB-CAN Interface. In Proceedings of the 2022 2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities Workshop (DI-CPS), Milan, Italy, 3–6 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 14–23. [CrossRef]
12. Buscemi, A. Automation of Controller Area Network Reverse Engineering: Approaches, Opportunities and Security Threats. Ph.D Thesis, University of Luxembourg, Luxembourg, 2022.
13. North IT Group GmbH. IT in the Automotive Industry: Transformation & Trends 2024 . 2024. Available online: https://northitgroup.com/en/knowledge/148-it-in-der-automobilindustrie-transformation-trends-2024 (accessed on 27 August 2024).
14. Rosenberger, N.; Rosner, P.; Bilfinger, P.; Schöberl, J.; Teichert, O.; Schneider, J.; Gamra, K.A.; Allgäuer, C.; Dietermann, B.; Schreiber, M.; et al. Quantifying the State of the Art of Electric Powertrains in Battery Electric Vehicles: Comprehensive Analysis of the Tesla Model 3 on the Vehicle Level. *World Electr. Veh. J.* **2024**, *15*, 268. [CrossRef]
15. Wassiliadis, N.; Steinsträter, M.; Schreiber, M.; Rosner, P.; Nicoletti, L.; Schmid, F.; Ank, M.; Teichert, O.; Wildfeuer, L.; Schneider, J.; et al. Quantifying the state of the art of electric powertrains in battery electric vehicles: Range, efficiency, and lifetime from component to system level of the Volkswagen ID.3. *eTransportation* **2022**, *12*, 100167. [CrossRef]
16. Rosenberger, N.; Deininger, S.; Koloch, J.; Lienkamp, M. Holistic Electric Powertrain Component Design for Battery Electric Vehicles in an Early Development Phase. *World Electr. Veh. J.* **2025**, *16*, 61. [CrossRef]
17. Oh, G.; Leblanc, D.J.; Peng, H. Vehicle Energy Dataset (VED), A Large-Scale Dataset for Vehicle Energy Consumption Research. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 3302–3312. [CrossRef]
18. Grunditz, E.A.; Thiringer, T. Performance analysis of current BEVs based on a comprehensive review of specifications. *IEEE Trans. Transp. Electrif.* **2016**, *2*, 270–289. [CrossRef]
19. Buscemi, A.; Turcanu, I.; Castignani, G.; Crunelle, R.; Engel, T. Poster: A Methodology for Semi-Automated CAN Bus Reverse Engineering. In Proceedings of the 2021 IEEE Vehicular Networking Conference (VNC), Ulm, Germany, 10–12 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 125–126. [CrossRef]
20. Marchetti, M.; Stabili, D. READ: Reverse Engineering of Automotive Data Frames. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1083–1097. [CrossRef]
21. Nolan, B.C.; Graham, S.; Mullins, B.; Kabban, C.S. Unsupervised Time Series Extraction from Controller Area Network Payloads. In Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, IL, USA, 27–30 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5. [CrossRef]
22. Ezeobi, U.; Olufowobi, H.; Young, C.; Zambreno, J.; Bloom, G. Reverse Engineering Controller Area Network Messages Using Unsupervised Machine Learning. *IEEE Consum. Electron. Mag.* **2022**, *11*, 50–56. [CrossRef]
23. Buscemi, A.; Turcanu, I.; Castignani, G.; Crunelle, R.; Engel, T. CANMatch: A Fully Automated Tool for CAN Bus Reverse Engineering Based on Frame Matching. *IEEE Trans. Veh. Technol.* **2021**, *70*, 12358–12373. [CrossRef]
24. Verma, M.E.; Bridges, R.A.; Sosnowski, J.J.; Hollifield, S.C.; Iannacone, M.D. CAN-D: A Modular Four-Step Pipeline for Comprehensively Decoding Controller Area Network Data. *IEEE Trans. Veh. Technol.* **2021**, *70*, 9685–9700. [CrossRef]
25. Verma, M.E.; Bridges, R.A.; Hollifield, S.C. ACTT: Automotive CAN Tokenization and Translation. In Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 13–15 December 2018; IEEE: Piscataway, NJ, USA, 2018; Volume 14, pp. 278–283. [CrossRef]
26. *Norm ISO 14229-1*; Road Vehicles—Unified Diagnostic Services (UDS). Part 1: Application Layer. International Standard: Geneva, Switzerland, 2020.
27. Rosenberger, N.; Wagner, N.; Fredl, A.; Riederle, L.; Lienkamp, M. Quantifying the State of the Art of Electric Powertrains in Battery Electric Vehicles: Comprehensive Analysis of the Two-Speed Transmission and 800 V Technology of the Porsche Taycan. *World Electr. Veh. J.* **2025**, *16*, 296. [CrossRef]
28. *Norm ISO 13400-4:2016*; Road Vehicles—Diagnostic Communication over Internet Protocol (DoIP). Part 4: Ethernet-Based High Speed Data Link Connector. International Standard: Geneva, Switzerland, 2016.

29. *Norm SAE J1962*; SAE J1962 Diagnostic Connector. Society of Automotive Engineers: Warrendale, PA, USA, 2016.
30. *Norm IEC 60603-7-2*; Connectors for Electronic Equipment–Part 7-2: Detail Specification for 8-Way, Unshielded, Free and Fixed Connectors, for Data Transmissions with Frequencies up to 100 MH. International Electrotechnical Commission: Geneva, Switzerland, 2010.
31. *Norm ANSI/TIA-568.0-E*; Generic Telecommunications Cabling for Customer Premises. American National Standards: Washington, DC, USA, 2020.
32. *Norm ISO 13400-3:2016*; Road Vehicles—Diagnostic Communication over Internet Protocol (DoIP). Part 3: Wired Vehicle Interface Based on IEEE 802.3. International Standard: Geneva, Switzerland, 2016.
33. *Norm ISO 13400-2:2019*; Road Vehicles—Diagnostic Communication over Internet Protocol (DoIP). Part 2: Transport Protocol and Network Layer Services. International Standard: Geneva, Switzerland, 2019.
34. Ye, W.; Zheng, G.; Cao, X.; Ma, Y.; Zhang, A. Spurious correlations in machine learning: A survey. *arXiv* **2024**, arXiv:2402.12715.
35. Rosenberger, N.; Lienkamp, M. Vehicle Parameter and Electric Powertrain Efficiency Analysis Using Real-Driving Data. *Int. J. Electr. Electron. Eng. Telecommun. (IJEETC)* **2024**, *13*, 494–502. [CrossRef]
36. GmbH, P. *Betriebsanleitung EV EasyCharge*, 3rd ed.; Pulsares Shop: Nienstädt, Germany, 2023.
37. Lawson, C.L.; Hanson, R.J. *Solving Least Squares Problems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1995.
38. Poslavskaya, E.; Korolev, A. Encoding categorical data: Is there yet anything'hotter'than one-hot encoding? *arXiv* **2023**, arXiv:2312.16930.
39. Naidu, G.; Zuva, T.; Sibanda, E.M. A review of evaluation metrics in machine learning algorithms. In *Artificial Intelligence Application in Networks and Systems, Proceedings of the 12th Computer Science Online Conference, Online, 2023.* Springer: Berlin/Heidelberg, Germany, 2023; pp. 15–25.