AI-based classification of CAN measurements for network and ECU identification

Ralf Lutchen1, Andreas Krätschmer1, Hans Christian Reuss1

Received: 27 September 2021 / Accepted: 29 July 2022 / Published online: 26 August 2022
© The Author(s) 2022

Abstract
Due to the constantly increasing number of functions offered by a modern vehicle, the complexity of vehicle development is also increasing as a result. A first indication of this connection is provided by the number of ECUs (electronic control units) used in current development vehicles. Furthermore, each ECU also performs more functions and is not only electrically networked with the other ECUs, but also logically and functionally. On this basis, new cooperative functions are being developed, which are used for example for autonomous driving. In vehicle development, more and more test sequences (diagnostic scripts) are established for function testing of individual components, systems and cross-functional methods. Due to decentralization and the modular approach, modern development vehicles consist of different numbers of ECUs. The high number of ECUs in purpose and number poses a challenge for test creation and updating. The ECU software is also developed in cycles within the vehicle cycle. This results in a very high software variance. This variance leads to the fact that in the vehicle development with global test conditions works. Global test conditions at this point mean that more ECUs are included in the measurement procedure than are installed in the vehicle. The vehicle structure (control unit and its software version) is not known to the person performing the measurement. He relies on the fact that his ECUs are inside in the global measurement task. This means that the vehicle network architecture is uncertain, which can lead to errors during test execution. Since the ECUs that are actually installed in the vehicle are first determined during test execution, this results in a longer script runtime than would be necessary. To support the development engineer and prevent avoidable errors, the diagnostic system should configure itself as far as possible. This means that individually customized measurements for each vehicle should be calculated in the cloud and not the global measurement tasks. For a diagnostic system to be able to configure itself independently, the vehicle network structure must be determined in a first step. This can be done by a simple CAN measurement (measurementXY.asc). An AI is able to analyze this measurement and classify the occurring ECUs as well as CAN networks. For larger measuring devices with more than one CAN interface, the user who analyzes the measurement is interested in which CAN was connected. Here, the AI is suitable to determine the name of the network and the communicating ECUs based on the communication that runs over the bus. For this purpose, the AI classifies the number of communicating ECUs based on the time intervals at which messages are sent. In addition, the AI can be supported by a special diagnostic script (global.pattern) to determine the vehicle structure at the OBD (on-board diagnostics) interface with maximum accuracy. Three AI approaches are presented, all connected in series and passing results to each other (pipeline mode). First comes the AI that separates vehicle communication from diagnostic communication. Based on the vehicle communication, the network name can be determined. Based on the diagnostic messages, the ECUs can be determined.

Keywords Artificial intelligence · ECU · IoT devices · Test sequences · Vehicle development

1 Motivation and research question
As shown in prior work [1], a modern vehicle is constantly receiving more functions and is, therefore, also more strongly networked. Each ECU gets and increasing amount of tasks and supports different standards [2]. Vehicle
networks are also changing from a structure of central gateways to a system of distributed gateways [3].

In the present work, an IoT diagnostic system is used, which offers many hardware interfaces as is common in vehicle development. The used system has 12 CAN bus interfaces, 2 FlexRay interfaces, 1 RJ45 automotive Ethernet and 1 BroadR-Reach interface, and this measurement system is connected to the cloud via a GSM sim card, WIFI or Bluetooth. This IoT measurement system can perform simple measurement tasks, such as listening to the CAN bus or FlexRay, but it can also perform more sophisticated measurement tasks. For example, a diagnostic script can clear the fault memories of the ECUs or a deterministic cyclic measurement (XCP (Universal Measurement and Calibration Protocol)-measurement) monitors a signal on a specific bus system. To create and execute a more sophisticated measurement task for a vehicle, some expert knowledge is needed. For example, the vehicle network architecture must be known, as well as the installed ECUs. In contrast to production vehicles, vehicles in development are more frequently modified or have prototype software versions. This poses certain challenges in vehicle development, making the use of AI algorithms a worthwhile alternative.

The aim of this work is to reliably detect the vehicle network architecture of the vehicle connected to the IoT diagnostic tester using appropriate AI algorithms. So that in a further step, it is possible to individualize the measurement tasks for this vehicle. For this purpose, different artificial intelligence algorithms are investigated and evaluated. For the results in this paper, two different series vehicles were used (a hybrid and a diesel vehicle), since it is primarily a matter of the general question of how the vehicle network structure can be determined with an AI.

Figure 1 shows the concept of the present work. A vehicle can be seen that is connected to an IoT diagnostic tester, which in turn is connected to a cloud. 2 There are two types

![Concept of the detection of the vehicle network architecture by AI algorithm](image)

**Fig. 1** Concept of the detection of the vehicle network architecture by AI algorithm

---

1 A typical example from the vehicle development: A person is to carry out vehicle measurements of a vehicle unknown to him, then in a first step the person must invest time to find out something about the vehicle and to be able to request the right files from the right colleagues. This expert-driven process makes it difficult to generate valid vehicle measurements.

2 In this use case, the cloud consists of several servers, which corresponds to a cluster that performs different tasks [15, 17, 19]. The applications running on the cloud cluster are classified as requiring protection due to their proximity to development, which means that the requirements of ISO 27001 must be met [13] and there is also room for experimentation with blockchain technology [9, 13, 20].
of files that are available in the cloud, on one hand configuration files such as diagnosis “.odx”, measurement task “.qtt” or bus systems “.arxml” and on the other hand, the measurements of the vehicle which can be, for example, an FCT (Full Can Trace) or a diagnostic script. The configuration files are encrypted and the keys are stored tamper-proof in the blockchain. The system-relevant data are thereby read out and stored in the corresponding database. When an FCT is executed in the vehicle and this has been transmitted to the cloud, the right path of the figure becomes active. The individual CAN networks measured are time-synchronized and combined to form an overall vehicle measurement. This measurement is subsequently transferred into a machine learning common panda data frame and stored in a “.pkl” file. This enables an AI to quickly read and process this file. When the machine learning file is computed from the measurement, the different trained models can now perform the respective classification and thus generate, for example, an ECU list as shown in the figure. The left path is traversed when a special diagnostic script is executed.

The research question is: How can a vehicle network architecture and all control units be reliably evaluated by an AI? This knowledge can be used, for example, to create individual measurement tasks. These measurement tasks can then be optimized to a minimum of complexity and execution time while guaranteeing completeness. In the following chapters, the generation of a data pipeline and the resulting classification tasks are described. In this paper, the supervised learning results of the classification algorithms [4–8] are presented. Also, in another paper, the results obtained with deep neural networks are presented.

2 Create a diagnostic pipeline to generate a machine learning dataset

To evaluate data with an AI, the data must first be recorded and processed. Data can then be used to train and analyze different AI models. The IoT diagnostic tester is used to record the data, as shown in Fig. 2. On the left side is the cloud and the AI algorithms, on the right side is the vehicle and in the middle is the IoT diagnostic tester with its hardware interfaces shown as a link. A special diagnostic script is started with a simultaneous FCT measurement on all connected CAN bus channels. In the case of the present measurement system, there is a limitation to a maximum of 12 CAN bus signals at the same time. The results are automatically uploaded to the cloud.

The diagnostic result is passed to a so-called pattern branch. In this branch, all ECUs are identified (name, variant, software version). The ODX standard is used as the data basis [1]. The individual CAN Bus channels are used to validate the pattern ECU verification. In a first step, the single measurements are merged to a complete file by the so-called FCT Merger.

This file is then read in and all the necessary steps to obtain a machine learning dataset are run through. This approach is based on the ARXML (Autosar Extensible Markup Language) standardization.

The diagnostic pipeline consists of an ARXML parser. This software module reads an ARXML file, extracts all relevant data and stores it in a global database. Thus, all ARXML files are automatically read when they are uploaded to the cloud server and the global database grows. Figure 3 shows three stages that must be passed in sequence to identify a vehicle network architecture. The first stage is the ARXML parser as just described. A short summary of the most important functions in the development of this module are also shown in Fig. 3.

In the analysis, neural networks [12, 12, 16] as well as recurrent neural networks (RNN) and reinforcement [20] approaches were investigated and compared.
Through this global database, several AI approaches can be tested in the following. On one hand, the information can be used to analyze a merged vehicle measurement based on the ECUs it contains. Again, it is possible to determine the vehicle architecture, because an ARXML always contains an image of an entire vehicle network. This means that in practice a swapped CAN bus would not be so dramatic as this error can be detected and corrected by the AI. A swapped CAN bus occurs for example when the engine CAN should be measured but in fact the transmission CAN was connected hardware-wise. For measurement systems without AI, a person must determine what is connected to an interface. With the approach presented here, the AI determines this itself.

The FCT Merger has the primary task of preparing a dataset for machine learning algorithms. That is, to make one file out of several “.asc” files. Here, the time axis must be considered and matched. The resulting file must now be read and processed in the Python programming language. Here, it is necessary to distinguish between two scenarios. First, the scenario in which the AI is already trained and the file must be classified. This is done in the following with the measurements of the test vehicle (diesel vehicle). The second scenario is that the AI must first be trained. This is done in the following with the measurements of the training vehicle (hybrid vehicle). Figure 3 shows the last stage, the FCT Merger, which again needs data from the stage above. With the data contained in the database “ECU_Classification_Data”, the data can be combined with classification results to then train supervised learning models. Target columns for two AI models are inserted. The first target is to distinguish a diagnostic line from a vehicle communication line. The second is to insert the sending ECU of each line.

### 2.1 ARXML parser

An ARXML file is an XML file in which a complete structure of a vehicle network is described. A vehicle network is, for example, a CAN bus to which different numbers of ECUs are connected. Some ECUs take over gateway tasks whereby messages are transmitted to other vehicle networks. A vehicle network consists of the combination of several Busses desript by ARXML files, which are connected via the gateway ECUs.

In the present work, the focus is not on the interpretation of the individual measured values, but on learning, the vehicle structure. From this aspect, the data of the individual ARXML files can be summarized in a global file (.csv or database table). This merging of the data results in three possible AI approaches.

First, an AI can learn which vehicle network is associated to a bus number in a measurement. This AI is named VCI (Vehicle Communication Interface) classification. By this approach, it is possible to automatically determine the correct and best fitting ARXML file from all vehicle networks.
that can occur in a vehicle at a vehicle manufacturer. Vehicles receive new features continuously during development, which naturally result in newer ARXML files. Thus, the first task of this AI is to recognize which vehicle network it is in general. As soon as this has been determined, the most suitable ARXML file must be selected based on the vehicle communication. This can of course also narrow down the software version of the ECUs.

Second, the “ECU group” can be determined based on which messages are transmitted on the bus. This AI is named ECU Group Classification. Unlike the ECU identification with vehicle diagnostics (ODX), the ARXML file describes only one CAN bus and its ECU function groups. This means that based on the ARXML file, it is only possible to determine the present of a certain group e.g. engine control unit. This is beneficial for generalization of AI approaches but has a negative impact on concrete ECU determination (diesel, hybrid, gasoline or electric).

As a last function, a distinction can be made between diagnostic messages and continuous vehicle communication. This AI is named DIAG classification, which makes it possible to separate the diagnostic messages from the vehicle messages in an FCT trace. This works on the example of the multi-master communication network CAN based on the identifiers that are sent in the arbitration phase. In general, important messages (e.g., “fire airbags”) have a low identifier. Since offboard diagnostic messages have a low priority, diagnostic messages tend to be found at the end of the 11-bit identifier (at higher numbers). The AI separates the two areas by threshold as will be shown in the following. For all these functions, the ARXML parser and the generation of an AI database is fundamental.

### 2.2 Python dataset

To show the individual AI approaches in the following, it is fundamental to create a methodology how a dataset can be developed with which models can be trained. The global ARXML database serves as the data basis for the target columns. Which was implemented for the following demonstration as “.csv” File. Through the ARXML file three classifications are implemented as shown in Fig. 4. The “BusNumber” is used to train the VCI classifier. Here, an individual number of the measurement is compared with the database and the corresponding number of the database is determined based on a similarity measure. The column “is_UDS” is used to train the DIAG classifier. The column “targets” represents the “ECU group” with which the ECU group classifier is to be trained.

Figure 4 shows the data originate from a vehicle measurement with a total of five connected CAN networks which are merged into one file by the FCT Merger.
the left part of the figure shows the column names (feature names) of the dataset and the data records (541,124 lines). A "asc" file usually contains more information than shown here, but this is purposely removed to reach a generalization. CAN, CANext and CAN-FD measurements are read in and prepared for AI classification in the same way. For a CAN-ext measurement, a 29-bit identifier is used instead of an 11-bit identifier. With a CAN-FD measurement, the frame area is transmitted with a CAN basic baud rate and the data area with an increased data baud rate. In addition, not only 8 bytes of user data are transmitted in the data area, but up to 64 bytes. As relevant feature names the Time, Id, DLC and the Payload Bytes are used to train the different AIs. The right side shows an excerpt from the CAN measurement. There are 20 data records shown and the data are available as integers. A transformation from the hexadecimal number system to the decimal number system has been performed.

To prove that an AI is better than random chance, the trained AI model must be tested on data that this model has never seen before. This is a test of the generalization performance of an approach. This is called a train–test split. The train–test split is implemented here based on two different vehicles. A hybrid vehicle with 41 ECUs is used for training and a diesel vehicle with 38 ECUs is used for testing.

Figure 5 shows the histogram plot [9] of the hybrid vehicle. The Time column is not relevant without further processing and will, therefore, be removed for future investigations. Furthermore, there is a restriction due to the measuring system used. In the case of diagnostic messages, only the received response is logged, the sent query is not logged. Furthermore, of the 4 possible can frames, only the data frames are received and logged. Error-, Overload- and Remote-frames are not logged.

Figure 5  Histogram of the data before cleaning

5 Furthermore, there is a restriction due to the measuring system used. In the case of diagnostic messages, only the received response is logged, the sent query is not logged. Furthermore, of the 4 possible can frames, only the data frames are received and logged. Error-, Overload- and Remote-frames are not logged.
was analyzed whether it concerns whole lines of zero values or individual values. Furthermore, it was examined whether an approximation with the mean or median improves the data set. The rows in which all PBs have the value zero were deleted as a result, since an approximation is not meaningful. Since the affected identifiers have no numerical scale in the interpretation. In addition, the removal of duplicates has been shown to be very worthwhile. Since in the course of the measurement, individual values do not change and thus a unique entry is sufficient. The reduction of the data set by removing the duplicates also has the advantage that the data set becomes smaller by a factor of 30. This is clearly noticeable in the runtime and resource utilization of the server in the cloud when training or classifying.

There are ECUs that make up a very large proportion of the total communication (engine, transmission, tank) and there are ECUs that have only responded to the diagnostic message (window regulator, air conditioning). From this unequal ratio of the information of the data, it is already clear that the accuracy as a classification metric will not suffice. Different metrics can be used for classification tasks. If a normal distribution can be seen in the histogram, the accuracy is used, in all other cases the metric must be worked on. In most cases, the data scientist gets an overview with the confusion matrix, which is also used here later to compare the AI approaches against each other. A simple example is provided by the classification of diagnostic messages. These are very much not equal distributed, as can be seen in the histogram (Is_UDS) of Fig. 5. There are a total of 41 pieces in 541,124 messages. This means that if an AI evaluates according to the accuracy metric, it will always say "it's not a diagnostic message" and thus arrive at a score of 99,9924232%. This should make it clear that based on the data, the metric chosen will make a difference in the performance of the trained AI models.

Figure 6 shows how the data set has changed after the adjustment. The feature time is deleted. 17,059 rows with only zero values in all Payload Bytes and 506,812 full duplicated Datapoints have been deleted. Thus, the dataset still has 17,253 rows which are all individual and can be

![Fig. 6 Histogram of the data after cleanup](image)
presented to an AI for training or classification. Furthermore, the distribution of the messages of the individual vehicle networks is highlighted under “BusNumber”, in Fig. 6. It is evident that most messages were transmitted in CAN Bus 10. The column “Target”, in Fig. 6 represents the 41 different ECUs and their proportion of the total communication. Only a few ECUs communicate a lot and that there is no equal distribution. The column "DLC", in Fig. 6 shows that almost every message in the diagnosed vehicle is 8-byte long. Therefore, this column will have no relevance for the AI but this cannot be generalized to all vehicles therefore this column remains.

The last column to be considered is the “Id” column. This is not independent of the “is_UDS” column. Very large identifiers are used for vehicle diagnostics. Figure 7 shows an enlarged version of the “Id” column”, of Fig. 6. Safety
critical functions use a low Id to get bus access as fast as possible and thus be able to put their information on the bus. The vehicle diagnostics is located in the upper third of the possible identifiers and can therefore be learned later by an AI with a linear approximation. The distribution of the Id is also relevant to determine the corresponding CAN network and consequently the ARXML file.

Finally, Fig. 8 shows the entire data set, which can now be created fully automatically by the data pipeline and made available for machine learning approaches. On the main diagonal are the already known histograms. In the other fields are the 17,253 individual data points colored in 41 colors each for the ECU classification. The separability is a difficult challenge which speaks for the use of a neural network. In the following nevertheless a classical approach is to be represented and the approach with the neural network comes in a later publication. In the following chapter, we will show how the different machine learning approaches prove themselves on the collected data set.

### 3 Classification

As already shown, the data set is to serve as a basis for three different AI models. The first use case is the detection of the connected CAN bus. This is equivalent to learning the Vehicle Communication Interface (VCI). With this methodology, the connected network should be identified. Thus, hardware errors (complete interchange of two CAN) become partially irrelevant since the assignment can be guaranteed on the Cloud. Furthermore, it is also irrelevant to which CAN bus the measuring system is connected. Thus, it does not have to be determined in advance in the cloud and is thus not a hard condition of the measurement. This AI helps to get closer to the goal of a completely self-configuring diagnostic process.

The second use case is the separation of the diagnostic messages from the vehicle communication. This detection of the diagnostic messages can be used to determine whether a diagnostic tester is on the bus or not. If the IoT diagnostic tester is the only offboard tester, it can communicate with the ECUs without hindrance. Due to its low priority, it interferes relatively little with vehicle communication. It only temporarily increases the bus load. If there is a second diagnostic tester in the development vehicle with an active diagnostic session, the IoT diagnostic tester could withdraw so as not to disturb this diagnostic session. Which, conversely, would lead to an abort of the action that is currently being actively executed. For example, if another diagnostic tester is already updating an ECU. This functionality is also interesting from a security point of view. Because during the update process of an ECU, the ECU is only available in a very limited way and furthermore a further diagnostic communication can lead to an abort of e.g. the flash function of the first diagnostic tester which puts the ECU into an undefined state. The consequence could be that this ECU must be removed and reloaded via a separate process (boot loader). In addition, by separating diagnostic and vehicle messages, further expert-driven AI approaches can be developed.

The third use case is the detection of an ECU group. An engine ECU has a standardized diagnostic request (e.g. 0×7E0) and response (e.g. 0×7E8) address. When an identification diagnostic message is sent, the ECU responds. This means on the level of ARXML and analysis of the vehicle communication, only the ECU group can be determined. The algorithm gets to know that an engine control unit is installed but not which one. However, this is sufficient to perform a validation of the diagnostic pattern approach.
Since the exact ECU with the associated variant and group was determined in the diagnostic pattern.

Building on these basic use cases, many more are conceivable and possible. Figure 9 shows the summary of the three use cases presented here. The figure contains three areas, one for each classifier. The three classifiers are discussed in more detail in the following chapters.

3.1 Vehicle communication interface (VCI)

When creating the global ARXML database, these files are read in, and unique IDs are assigned for the bus systems and files read in. In the case of a CAN measurement, the measuring system also reports a number for the channel on which the data was received. Now, the two numbers must be combined. Currently, an engineer must indicate which CAN bus was connected after the measurement. In the future, the connected vehicle network will be automatically identified by this AI and the possible ECUs will be identified based on this. Furthermore, an engineer must specify which ARXML file should be used to further process the measurement. The AI will also simplify this step in the future by determining and applying the best fitting (most complete) ARXML file ID from the database.

In the case of network analysis, it is important to evaluate the history of the messages as the low identifiers in each CAN network are occupied. Another possibility is to extract the diagnostic messages of the analyzed CAN network and to perform the network recognition based on these messages. This takes advantage of the fact that different ECU groups are operated on different vehicle networks, but all groups must be unique again in the vehicle.

Since this model can only be realized by combining another model and even then, still requires a recurrent neural network, a more detailed description is postponed to a future paper.

3.2 Diagnosis messages

This model has the task of separating the diagnostic messages from the vehicle communication. In the train data set, there are 41 ECUs, each of which has transmitted a diagnostic message. In the test data set, there are 38 ECUs (another vehicle is used as the test data set).

In Fig. 10, the results of three approaches are compared. Due to the distribution, the Confusion Matrix is used here. The accuracy cannot be considered as a metric at this point, because the data are not normally distributed (17,212 vehicle communications to 41 diagnostic messages). The three classifiers shown here are:

- on the left side, the SVM (support vector machine) with linear kernel algorithm
- on the right side, the k-nearest Neighbors algorithm
- in the middle, the random forest algorithm

The results are different, as shown in Fig. 8, the data can be approximated by a straight approximation, which also gives the best results for the SVM with linear kernel. The Random Forest and the k-nearest Neighbor classifier go into overfitting for a given data set. This can be clearly seen here,
as the classifier works very well on the trained vehicle but produces only very unsatisfactory results on a new vehicle. Thus, only the SVM achieves the generalization for this task.

3.3 ECU

The task of this model is to assign the most probable ECU group to each row in the data set. There are two different vehicles in the train–test split. As already seen in the histograms, there are some ECUs with a very large communication share and again ECUs with only one message. This condition and the fact that there is no clear criterion by which the data can be separated. This leads to ambiguous results in the following Confusion Matrix consideration.

Different algorithms were examined. Figure 11 shows that the algorithm SVM with linear kernel does not provide reliable results. The model is too simple and cannot learn the complex data. It has also been shown that increasing the database does not give satisfactory results. Thus, the model must become more complex.

Figure 12 shows the results of the k-nearest neighbors algorithm. Here, too, the desired goal cannot be achieved due to the data set. A full occupation of the main diagonals is expected. Neither the k-nearest Neighbors nor the SVM algorithm achieve a result that is better than the random probability for this task without adjusting the data set. In the case of the k-nearest Neighbors algorithm, the cause is most likely due to the fact that there are representatives with only
one message due to the data set and the algorithm has too few neighbors to function meaningfully.

Figure 13 shows the results of the random forest algorithm. Despite the data set in which an ECU group occurs only once, it can still be identified. Thus, the random forest is in principle suitable for this task. However, the data basis would have to be increased by additional vehicles to achieve more valid results.

Alternatively, a change to a more complex model is possible. Here, a neuronal network [10] would be advantageous because learning in epochs can partially compensate for the disadvantage of the uneven distribution of messages.

### 4 Summary/outlook

In the course of this work, a complete data pipeline was developed. This makes it possible to prepare arbitrary CAN measurements for further processing with a machine learning model. The developed data pipeline automates the process of ECU validation of the diagnostic pattern in the cloud, which is an essential step in terms of automatic test generation.

One machine learning model (LinearSVM) can separate diagnostic messages from vehicle communication in a very simple way. Another model (Random Forest) is suitable in principle for assigning the most probable ECU group to each line in an “.asc”-file. This model is intended to support and validate the diagnostic pattern approach.
The high number of features could be further reduced by a main component decomposition. Investigations regarding computational performance and for batch learning in the cloud are useful and will be investigated.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

**References**

1. R. Lutchen, A. Krätschmer und H. C. Reuss, „Concept for the Automatic Generation of Individual Test Sequences Verified by Artificial Intelligence Algorithms.“ in 21. Internationales Stuttgarter Symposium, Stuttgart, Springer Fachmedien Wiesbaden, 2021.
2. Zimmermann, W., Schmidgall, R.: Bussysteme in der Fahrzeugtechnik, Protokolle, Standards und Softwarearchitektur. Springer Vieweg, Stuttgart (2014)
3. Reuss, P.D.-I.H.-C.: Embedded controller & datennetze im kraftfahrzeug. Skript zur Vorlesung an der Universität Stuttgart, Stuttgart (2020)
4. Nolting, M.: Künstliche intelligenz in der automobilindustrie. Springer Vieweg, Hannover (2021)
5. Müller, A.C., Guido, S.: Einführung in machine learning mit python. O’Reilly, Berlin (2017)
6. Gruhn, V., Hayn, A.V.: KI verändert die Spielregeln. Carl Hanser, München (2020)
7. Frochte, J.: Maschinelles Lernen—Grundlagen und Algorithmen in python. Carl Hanser, Bochum (2018)
8. Ernst, J., Kaiser, P.: Python 3. Rheinwerk, Bonn (2017)
9. Aust, H.: Das Zeitalter der Daten. Springer Vieweg, Bonn (2021)
10. Rashid, T.: Neurionale Netze selbst programmieren. O’Reilly, Heidelberg (2017)
11. Schacht, S., Lanquillon, C.: Blockchain und maschinelles Lernen. Springer Vieweg, Heilbronn (2019)
12. Géron, A.: Praxiseinstieg maschine learning mit Scikit-Learn, Keras und TensorFlow. O’Reilly, Heidelberg (2020)
13. Fertig, T., Schütz, A.: Blockchain für entwickler. In: Würzburg. Rheinwerk, Bonn (2019)
14. Kloep, P.: Sichere windows-Infrastrukturen. Rheinwerk, Bonn (2020)
15. Kloep, P., Weigel, K., Momber, K., Rojas, R., Frankl, A.: Windows server 2019. Rheinwerk, Bonn (2019)
16. Lämmel, U., Cleve, J.: Künstliche Intelligenz—Wissensverarbeitung Neuronale Netze. Carl Hanser, Wismar (2020)
17. Liebel, O.: Skalierbare container-Infrastrukturen. Rheinwerk, Bonn (2021)
18. Stender, D.: Cloud-infrastrukturen. Rheinwerk, Bonn (2020)
19. Zai, A., Brown, B.: Einstieg in deep reinforcement learning. Carl Hanser, Hasselbach (2020)
20. Freiknecht, J., Papp, S.: Big data in der praxis. Carl Hanser, München (2018)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.