CANLite: Anomaly Detection in Controller Area Networks with Multitask Learning

Prashanth Balaji and Majid Ghaderi
University of Calgary
{prashanth.balaji, mghaderi}@ucalgary.ca

Hongwen Zhang
Wedge Networks
hongwen.zhang@wedgenetworks.com

***Abstract*—The Controller Area Network (CAN) bus has been a widely implemented standard for in-vehicle communication between vehicle subsystems. However, since CAN was never designed with a focus on security, attackers can exploit the lack of message authentication in CAN to inject crafted malicious payloads to disable critical systems onboard the vehicle. While previous works in literature focus on detecting deviations in the normal behavior of the bus, they merely focus on individual sensors. Hence they fail to identify stealthy attacks that do not cause individual sensors to deviate substantially from their expected behavior but still have a significant impact on the bus state. Further, such approaches often impose a computational strain on the deployed system due to the high magnitude of consumed resources at run-time. To this end, we propose CANLite, a lightweight anomaly detection system utilizing multitask learning to detect such subtle deviations while significantly reducing the memory footprint. We trained and evaluated our model against a state-of-the-art baseline approach. Our results indicate that CANLite reduces the memory footprint by 50% while still achieving the same level of detection performance as the baseline.**

I. Introduction

Recent advancements in the field of automotive electronics has enabled modern cars to interact and be more aware of their surroundings. At present, vehicles can perform a plethora of functions simultaneously to ensure the safety of passengers and also those around them. These range from tasks such as warning the driver of an unlocked door to complex actions such as semi-autonomous driving. The responsibility of reliably executing such advanced behaviors falls onto the Electronic Control Units (ECUs) that drive the corresponding actuators based on the information retrieved from the sensors in the vehicle. The ECUs are interconnected through the Controller Area Network (CAN) bus through which data points are exchanged in real-time. The CAN bus offers a fault-tolerant medium to effectively relay time-critical data and hence has been the de-facto standard for in-vehicle communication networks over the past two decades [1]. However, CAN lacks basic security features such as authentication and encryption to safeguard vehicular subsystems from attackers trying to gain unauthorized access. Historically, in-vehicular networks were typically isolated and therefore executing an attack required physical access to the vehicle [2]. But currently, the incorporation of complex wireless technologies such as C-V2X and Bluetooth has provided a wider attack surface for malicious actors to leverage. As such, an attacker can remotely monitor or execute commands to inject malicious payloads that could possibly disable critical systems while the vehicle is in operation. Security researchers in [3]–[5] demonstrated that an adversary would be able to successfully compromise an ECU through remote firmware updates and disable subsystems such as the electronic stability program and anti-lock braking. Therefore, a prominent method followed by automotive manufacturers to safeguard the bus is to isolate it from modules that interact with external networks and interfaces. However, an internal access port is typically present to manage data flows and therefore such defense mechanisms do not render the bus impenetrable.

While some attacks including DoS and fuzzy attacks on the bus can be identified by analyzing the traffic frequencies [6] or entropy [7], an attacker can specifically craft payloads while adhering to the system specifications and thus evade detection. Such attacks will be more difficult to detect due to their subtle nature and as such require a collective evaluation of the bus state during detection. Additionally, the applied payload encoding substantially increases the complexity as traditional detection techniques rely on decoded signal values that require the translation rules from the manufacturer. Although there exist numerous works in literature that utilize deep learning for anomaly detection in the CAN bus at a frame-level [2], [8], [9], they fail to consider this particular attack scenario. In [10], the authors focus on detecting such stealthy attacks by exploiting the sensor correlation between data points. They develop an individual neural network for every identifier and deploy them on the CAN gateway. However, at run time, this leads to managing a large number of models in memory. Automotive ECUs are embedded devices with limited computational resources and therefore such an approach might impose a huge computational strain on the deployed node, thus affecting the performance of the detection system. To this end, we present CANLite, a lightweight deep learning-based detection model that utilizes multitask learning to effectively curtail the high memory footprint while not compromising on detection accuracy. CANLite operates on raw CAN frames and learns the correlations between sensor datapoints to effectively detect stealthy attacks on the bus. To the best of our knowledge, CANLite is the first detection system designed for the CAN bus to utilize multitask learning on a frame level. In addition, the shared knowledge model of CANLite allows to localize the source of the anomaly once detected.

This work was supported by Wedge Networks Inc., Alberta Innovates and Natural Sciences and Engineering Research Council of Canada.

We summarize the main contributions of this work as follows:

1) We present the design of CANLite, a lightweight deep learning-based detection model that exploits inter-sensor relationships in raw CAN frames to identify stealthy attacks in an automotive CAN bus.
2) We implement and train CANLite over two datasets comprising raw CAN frames and compare it against a recently proposed approach called NeuroCAN [10].
3) We present a brief analysis of the corresponding memory footprints and inference times to substantiate the benefits of CANLite. Our results indicate that CANLite consumes only half the size in memory while achieving the same level of detection accuracy as NeuroCAN [10] with lower number of false negatives.

The remaining sections of this paper are organized as follows. Section II summarizes the related works. In Section III we provide an overview of the CAN protocol and the necessary machine learning background. We present the architecture of CANLite in Section IV. Section V discusses our experimental setup and results. Finally, Section VI concludes the paper.

II. Related Works

In this section, we provide a summary of a few closely related works that utilize deep learning for anomaly detection on the CAN bus. We categorize them into frame level and signal level detection approaches based on the type of data used during the detection process, i.e., either raw CAN frames or decoded signal values.

Frame Level Detection. The authors in [9] proposed a defense platform for anomaly detection on the CAN bus. Further, they presented an LSTM-based multiclass classification model deployed as part of the platform. The network was trained and evaluated to distinguish DoS, fuzzy and spoofing attacks. A network utilizing LSTM to predict the bit sequence in the ECU payload at subsequent time steps was presented in [2]. The authors evaluated their model over several types of data modification attacks and anomaly scoring approaches. In NeuroCAN [10], the authors leveraged the spatio-temporal relationships between multiple sensors to detect stealthy attacks on the bus. The proposed model utilized a combination of linear embeddings and LSTM to predict the payload sequence of an ECU in the next time step and achieved over 95% accuracy. In CANTransfer [1], the authors adopted transfer learning with convolutional LSTMs for identifying DoS attacks. The incorporation of transfer learning enabled the network to identify new attack scenarios while trained to only identify a single attack type. A two-stage detection model with Generative Adversarial Networks (GAN) was proposed in [11] for detecting DoS and fuzzy attacks. The network comprised two discriminators where during detection, the first was used for identifying known attacks while the other focused on detecting zero-day exploits.

Signal Level Detection. A distributed anomaly detection framework utilizing GRU-based autoencoders was proposed in [12]. The system primarily acts on decoded signal values with a specific focus on detecting subtle changes in the CAN payload caused by replay and injection attacks. In CANet [8], a standard autoencoder with LSTM embeddings was used to detect flooding, replay and data modification attacks. The authors in [13] jointly utilized an autoencoder and predictor model to develop a multitask anomaly detection system. The joint architecture comprised a single encoder and two decoders with convolutional layers and LSTM. The prediction model determined the next driving maneuver which was utilized by the autoencoder to better learn the input representation. The system was trained and evaluated over finding outliers in six signals in the payload. In [14], the authors presented a standard autoencoder and evaluated it over 68 sensor data points in the AEGIS big data project [15]. Further, different reconstruction metrics such as Mean Square Error (MSE), Mean Absolute Error (MAE) and Mean Square Logarithmic Error (MSLE) were adopted to analyze the robustness of the model.

III. Background on CAN and ML

A. Controller Area Networks

CAN is a serial bus communication protocol that was introduced primarily to provide a reliable medium for onboard vehicular control units to exchange data in real-time. Most modern vehicles have two CAN topologies with varying bandwidths, where the low-speed bus is utilized for simple functions such as indicators for door closure while the high-speed bus is utilized to manage time-critical applications such as airbag deployment. At the physical layer, CAN is implemented by means of two wires, *CAN high* and *CAN low* that drive the state of the bus to either a 1 or 0. These states have been termed as recessive and dominant in the CAN standard [16]. CAN enables the exchange of messages between ECUs by encapsulating the payload within a data frame. The format of a standard CAN frame is illustrated in Fig. 1. We specifically describe the identifier and data fields in the frame as the rest are outside the scope of this paper. CAN is a broadcast protocol where data sent by a node is received by others actively connected to the bus. During a transmission cycle, the 11 bit CAN ID acts as a priority indicator such that the ECU sending the lowest value identifier wins control of the bus. An ECU transmitting on the bus is also capable of monitoring the state of the bus and thus withdraws from contention if it notices a lower value ID being sent. Once the payload is sent, proper reception is indicated to the sender by driving the bus to the dominant state during the acknowledgment cycle. If an error is noticed, error frames are transmitted on the bus to destroy the content of the transmitted frame. The data field in a CAN frame has a maximum size of 8 bytes comprising various signal values encoded through custom rules defined by the manufacturer. These rules define the bit boundaries of the values contained in the payload and are proprietary. Therefore, traditional detection approaches require them as they typically act on decoded signal values. However, CANLite operates at the frame level and hence does

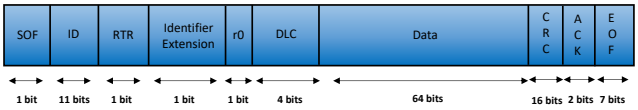


Fig. 1: Format of a Standard CAN frame

not require any prior knowledge of the signals contained in the payload.

B. Machine Learning

Long Short-Term Memory. First introduced by Hochreiter and Schmidhuber, Long Short-Term Memory (LSTM) [17] is a neural network architecture that is capable of retaining temporal information over longer periods. Prior to LSTM, Recurrent Neural Networks (RNN) were essentially used to perform time series forecasting. RNN maintains hidden states that allow the network to persist parts of the previous input, thus introducing the notion of time. During training, backpropagation occurs over a series of time steps causing the gradients to either shrink or explode when performed for longer sequences. On the other hand, an LSTM cell overcomes the problem of vanishing or exploding gradients by introducing a cell state along with the hidden state to regulate the amount of information persisted over time. As such, it allows the network to remember long-term dependencies while effectively reducing the influence of short-term memory.

Multi-Task Learning. Inspired by the human ability to leverage the knowledge gained over learning a task to another, Multi-Task Learning (MTL) is a paradigm in machine learning where a model is trained to jointly learn more than a single task. MTL strives to generalize the performance of the model over several related tasks such that multiple models are not trained from scratch for every targetted task. A reason for MTL being more effective than single-task learning is attributed to the additional data from learning other tasks that the model could utilize at the time of inference [18]. In deep learning, MTL is achieved either through hard parameter or soft parameter sharing [19]. Hard parameter sharing models comprise a shared segment in the network that effectively acts as a shared knowledge base for all the trained tasks. On the contrary, a separate model is developed for every task in soft parameter sharing while the distance between model weights is reduced during training. Further, the weights are updated by deriving the gradients over the combined loss of all tasks. Additionally, jointly learning multiple tasks reduces overfitting as it introduces a degree of inductive bias that causes the model to prefer a hypothesis over others.

IV. CANLite Architecture

CANLite is a lightweight anomaly detection model that primarily focuses on detecting stealthy attacks by identifying contextual anomalies in the CAN bus. To this end, the network architecture is fundamentally structured to capture both the spatial and temporal relationships between multiple sensors. The architecture of CANLite encompasses three segments, each with a distinct role to play in the detection process. The

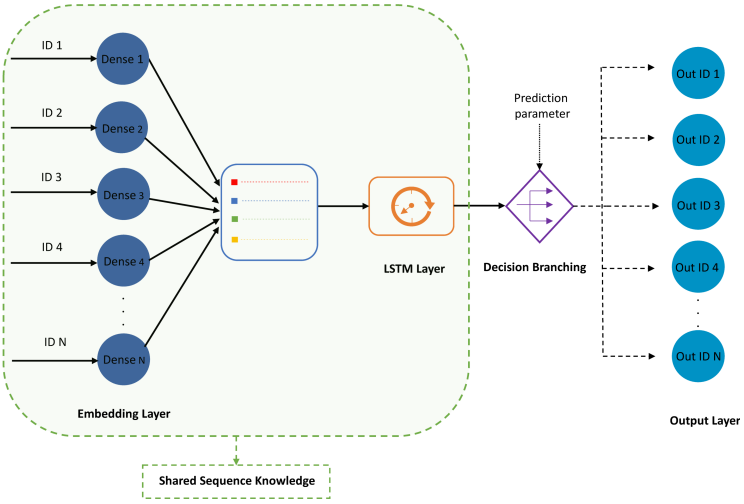


Fig. 2: Structure of CANLite.

structure of CANLite is shown in Fig. 2. In particular, the network structure comprises an embedding layer, an LSTM layer and then finally an output layer that specifically predicts the next payload sequence of the requested CAN ID. As the payload size of every identifier varies depending on the particular implementation by the manufacturer, the input to the network is first processed by the embedding layer to provide a fixed size feature set. Specifically, the embedding layer applies a linear transformation over the provided input and focuses on learning the data patterns within a payload from a single control unit. The second segment of the model is an LSTM layer that processes the cumulated embeddings to effectively learn the variations in sensor readings occurring over time. We utilize a standard LSTM cell with `sigmoid` and `tanh` activations and a hidden size of 32. The final segment of the network is an output layer comprising a set of linear units, one for every CAN ID whose payload the model is trained to predict. The output from the LSTM layer is passed to the branching module that channels it to the corresponding output unit based on the decision parameter. The decision parameter is essentially the CAN identifier for which the model has to predict the next data sequence. In particular, given a set of N identifiers, $I = \{ID_1, ID_2, ID_3, ..., ID_N\}$ with an associated time series $\{X_{i1}, X_{i2}, \ldots X_{in}\}$ for each ID_i, the network predicts $X_{i(t+1)}$ given X_{it}, W_{it}, Y_{it} and Z_{it}. Here, W_{it}, Y_{it}, Z_{it} are the data payloads transmitted by other nodes between timesteps t and $t+1$ and provides the necessary context for the model to collectively assess the state of the various nodes on the bus. The directed output unit then applies a linear transformation over the received input from the LSTM layer and finally scales it through the sigmoid activation function yielding values between 0 and 1.

V. Experiments and Results

In this section, we first provide an overview of the utilized datasets and the applied preprocessing. Later, we present an

TABLE I: Dataset overview.

Dataset	Messages	Normal	Injected
Gear dataset	4,443,142	3,845,890	597,252
RPM dataset	4,621,702	3,966,805	654,897

outline of our training setup and finally discuss our experimental results.

A. Dataset Specification

We trained and evaluated our neural network model on two datasets comprising raw CAN frames logged through the OBD port. The datasets were developed by The Hacking and Countermeasure Research Lab, Korea [11] and were acquired upon request. Fabricated CAN messages were simultaneously injected into the bus while capturing the network traffic. Specifically, the gear module and rpm gauge were targeted and frames containing the modified data fields were injected. Table I provides an overview of each dataset. We specifically focus on the data field of the CAN frame as we intend to train our model to learn the inter-sensor relationships between signal values encompassed in the payload. Prior to training, each dataset was split, with 70% allocated for training and the rest for testing. Further, the data fields were extracted and all hexadecimal values were converted and normalized such that each input feature represented one byte in the CAN payload.

B. CANLite Training

We jointly train the network over the two datasets such that at every timestep the model predicts the upcoming sequence of values expected in the data payload of a CAN ID. Further, to ensure that one task does not influence the model weights in the shared portion of the network, we combine the individual losses at every output unit and effectively derive the gradients over the cumulative error score at every forward pass of data. We trained our model only with normal CAN frames and utilized mini-batch training with a batch size of one. The smaller batch size can be attributed to the varying context size between timesteps and therefore is not feasible to combine over a larger input batch. The Adam optimizer [20] was utilized for training with the learning rate set to 0.001. For our experiments, we implemented our model with `PyTorch` [21]. We performed the necessary data preprocessing and performance evaluation with `pandas` [22] and `scikit-learn` [23]. The training was performed on a Linux CentOS 8 machine with a Tesla V100 GPU and 32 GB of memory.

C. Anomaly Score and Performance Measure

In our approach, we compared the network prediction for every sample in the test set against the expected data sequence and derived an anomaly score through Mean Square Error (MSE) given by

$$\mathrm{MSE}(X, \hat{X}) = 1/k \sum_{i=1}^{k} (x_i - \hat{x_i})^2 \,. \tag{1}$$

The performance of an anomaly detection system is highly sensitive to the decision threshold that allows to effectively segregate the anomalous samples such that the system achieves a high true positive rate. We determined that the optimal threshold for every CAN ID resulted in being the one with the maximum F1-score. The F1-score is the harmonic mean between the precision and recall and provides an accurate measure on the performance of a machine learning model. Further, we measured the detection performance of our network through the Receiver Operating Characteristic (ROC) curve.

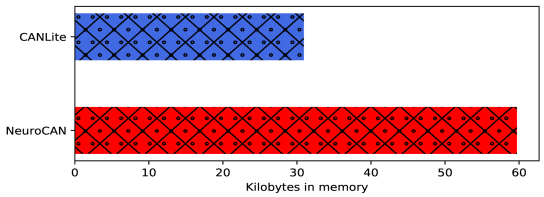


Fig. 3: Memory footprint at inference time.

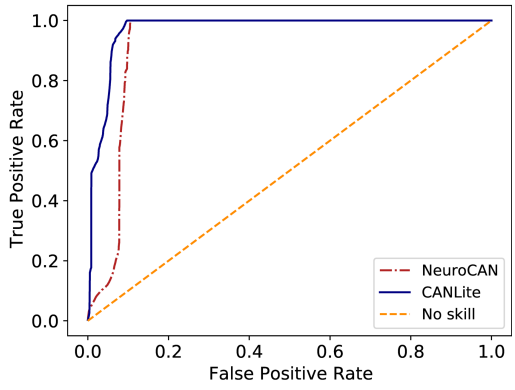
D. Baseline Model

In this paper, we present a lightweight detection model that primarily focuses on identifying contextual anomalies on the CAN bus. To highlight the performance of our approach, we compare it with NeuroCAN [10] that serves as our baseline. Similar to CANLite, the baseline also strives to identify contextual anomalies in CAN bus data at a frame level. However, in [10], a separate model was trained for every targeted identifier leading to a large number of detection models to be held in memory by the system at the time of inference. Furthermore, developing every model from scratch also imposes a higher computation cost while significantly increasing the time required for training.

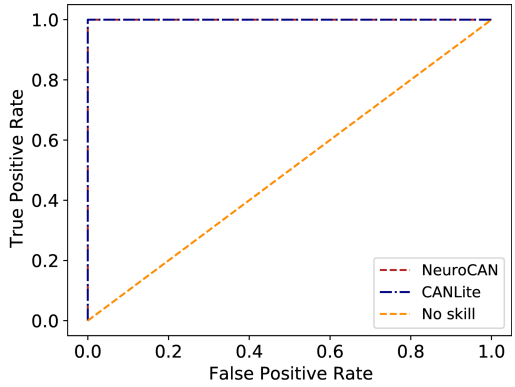
E. Results and Discussion

Memory Footprint. We determined the memory footprint of both models at the time of inference to comprehend the posed overhead on the deployed node on the bus. Owing to the shared learning between the embedding and LSTM layer in CANLite, it only consumes half the space taken by NeuroCAN, as depicted in Fig. 3. On evaluation, we observed that CANLite consumed 30.91 KB while NeuroCAN consumed 59.71 KB in memory. The higher footprint under the baseline approach is due to loading all the trained models into memory for the detection system to scan the appropriate frames while the vehicle is in operation. As such, it poses a significant computation overhead to achieve a high level of performance while meeting the necessary inference deadlines. On the other hand, our experimental results establish that CANLite achieves the same level of performance while reducing the unnecessary memory overhead, as presented in the next sub-section.

Detection Performance. The ROC curves obtained over evaluation against the Gear and RPM datasets are presented in Figs. 4(a) and 4(b). In comparison with the baseline, we achieve lower false-negative and false-positive rates while maintaining the detection accuracy with the Gear dataset. As such, CANLite achieves a lower rate of misclassification over the baseline for normal CAN traces thereby considerably reducing the number of false alarms. With the RPM dataset,



(a) Gear dataset.



(b) RPM dataset.

Fig. 4: Comparison of detection performance.

CANLite does not experience any degradation in performance when compared against the baseline and is able to successfully differentiate all injected sequences from normal CAN traffic. We summarize the detection measures along with the F1-score of both models in Table II.

Anomaly Localization. Once an anomaly has been identified, the detection system can then perform the necessary maneuvers to atleast warn the driver of the observed misbehavior. While previous approaches utilizing autoencoders [12] [14] do offer detection models that can substantially identify outliers at a frame level, they fail to localize the malicious node on the network. The malicious node could then potentially continue to transmit until the vehicle is completely stalled. However, CANLite approach of evaluating an ECU individually while analyzing the collective state of the bus allows to identify the malicious node by associating it with the corresponding CAN identifier. At the time of inference, we observed that CANLite was able to extract misbehaving node identifiers `043f` and `0316`, which are the CAN IDs in the injected anomalous frames.

VI. Conclusion

The increasing number of connected vehicles in recent years has facilitated the need to secure onboard vehicular systems such as the CAN bus to ensure the safety of all road users. In this paper, we propose CANLite, a lightweight deep learning-based detection model to identify stealthy attacks in CAN bus data. Our results indicate that CANLite achieves over 95% detection accuracy while imposing half the resource overhead posed by the existing approaches. Further, CANLite also facilitates in localizing the source of an anomaly thus enabling the system to reduce the impact of the malicious node when detected. In our future work, we plan to subject our model to incremental training as repeated learning would be necessary to account for the changes in vehicle behavior over time and also enable addition of new nodes to the bus.

TABLE II: Evaluation Results.

Dataset	Model	TPR	FPR	FNR	TNR	F1 score
Gear	CANLite	0.91	0.0001	0.088	0.99	0.9541
	NeuroCAN	0.90	0.00065	0.093	0.99	0.9508
RPM	CANLite	1.0	0.0	0.0	1.0	1.0
	NeuroCAN	1.0	0.0	0.0	1.0	1.0

References

[1] S. Tariq, S. Lee, and S. S. Woo, "CANTransfer: transfer learning based intrusion detection on a Controller Area Network using convolutional LSTM network," in *Proc. ACM SIGAPP*, 2020.

[2] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with Long Short-Term Memory Networks," in *Proc. IEEE DSAA*, 2016.

[3] S. Checkoway, D. McCoy, Kantor *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. USENIX Security Symposium*, vol. 4, 2011.

[4] H. J. Jo, W. Choi, Na *et al.*, "Vulnerabilities of android os-based telematics system," *Wireless Personal Communications*, vol. 92, 2017.

[5] Y. D. Sen Nie, Ling Liu, "Free-fall: Hacking tesla from wireless to CAN bus," in *Proc. Blackhat*, 2017.

[6] C. Young, H. Olufowobi *et al.*, "Automotive intrusion detection based on constant CAN message frequencies across vehicle driving modes," in *Proc. ACM Workshop on Automotive Cybersecurity*, 2019.

[7] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proc. IEEE Intelligent Vehicles Symposium*, 2011.

[8] M. Hanselmann, T. Strauss, Dormann *et al.*, "CANet: an unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, 2020.

[9] M. D. Hossain, H. Inoue *et al.*, "LSTM-based Intrusion Detection System for in-vehicle CAN bus communications," *IEEE Access*, vol. 8, 2020.

[10] P. Balaji and M. Ghaderi, "NeuroCAN: Contextual anomaly detection in Controller Area Networks," in *Proc. 2021 ISC2*, 2021.

[11] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proc. PST*, 2018.

[12] V. K. Kukkala, Thiruloga *et al.*, "INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems," *arXiv*, 2020.

[13] V. Sadhu, T. Misu *et al.*, "Deep multi-task learning for anomalous driving detection using CAN bus scalar sensor data," *arXiv*, 2019.

[14] T. He, L. Zhang *et al.*, "Exploring inherent sensor redundancy for automotive anomaly detection," in *Proc. ACM/IEEE DAC*, 2020.

[15] C. Kaiser, A. Stocker, and A. Festl, "Automotive CAN bus data," Jul. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3267184

[16] R. Bosch *et al.*, "CAN specification version 2.0," *Robert Bousch GmbH, Postfach*, vol. 300240, 1991.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv*, 2017.

[19] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv*, 2017.

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

[21] A. Paszke, S. Gross, Massa *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019.

[22] The pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[23] L. Buitinck, G. Louppe *et al.*, "Scikit-learn: Machine learning in Python," *JMLR*, vol. 12, 2011.