Fingerprinting Automobiles with CAN Bus Data Snapshots

David R Crow, 2d Lt, USAF

***Abstract*—**

***Index Terms*—classification, siamese neural networks, one-shot learning, CAN bus**

I. Introduction

MODERN vehicle manufacturers tend to employ a policy of *security through obscurity* when designing the controller area network (CAN) bus and its related components. However, this security policy is insufficient for consumer protection: obfuscating a vehicle's CAN data does not adequately hide the vehicle's signature. In other words, today's toolsets can successfully determine whether a specific vehicle generated a specific segment of CAN data, even if said data is unprocessed or limited in scope. This research presents two such techniques, both in the field of deep learning.

Specifically, we devise two systems capable of identifying which distinct vehicle generated a given segment of CAN bus data. This is a multiclass classification problem which asks the following question: does a given vehicle generate data with some characteristic unique to that vehicle? In other words, does a given vehicle leave identifiable fingerprints on its data?

We hypothesize that a siamese neural network (SNN) can effectively fingerprint vehicles. This is a relatively new deep learning technique for object classification, and it is primarily used in face recognition. When applied correctly, this method can learn to identify or distinguish pictures of an individual's face—even when given just a few training examples. In this research, vehicles are akin to faces; likewise, segments of captured vehicle data are akin to pictures of faces.

This research employs two datasets, one from Oak Ridge National Laboratory (ORNL) and one from Captain Brent Stone's doctoral research efforts. The lab shared nearly 2.5 gigabytes of data captured on the CAN buses of nine different vehicles; Captain Stone captured over 230 megabytes of data from 11 different vehicles. In this research, 1,024 bytes of sequential CAN data constitute one data snapshot, so formatting the datasets gives nearly three hundred thousand observations. We label every snapshot with the vehicle that generated the snapshot's data, so the deep learning task is fully supervised.

The remainder of this report presents the research in detail. Section II examines some of the related work in current literature and explains why this work is insufficient for the research at hand. Section III describes the dataset, the deep learning model, the model-fitting process, and the model analysis and evaluation tools. Section IV presents and discusses the results obtained. Section V explains the implication of the results and suggests possible opportunities for future research.

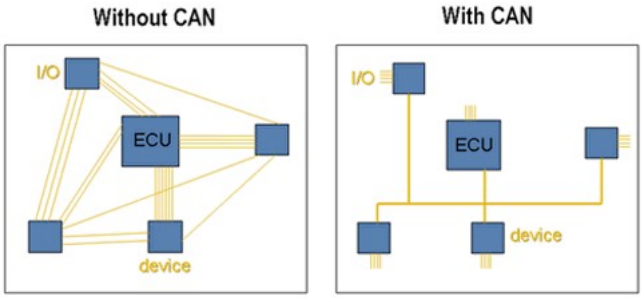

Fig. 1. Vehicle network wiring with and without CAN

II. Background & Related Work

To better understand this research, one must understand basic information about controller area networks, about deep learning, and about related research. This section provides this background information. Specifically, Section II-A discusses the controller area network, including its message structure and contents. Section II-B discusses time series classification approaches in deep learning and how they relate to this research. Section II-C discusses related work in CAN analysis and fingerprinting vehicles, both with and without deep learning.

A. The Controller Area Network

The controller area network is a message broadcast system developed for automobile applications by Bosch in the early 1980s. The CAN broadcasts short messages detailing the vehicle's functionality (e.g., wheel speed, engine temperature, velocity); this ensures data consistency throughout the vehicle. As Figure 1 shows, this also reduces required wiring [1].

CAN messages are limited in capacity: each one contains message overhead—including an arbitration ID, which identifies the message's contents to the network—and up to 64 bits of data [2], [3]. Additionally, the components along the CAN do not possess a wealth of computing power, so computational capabalities are also limited. For this reason, automobile manufacturers typically employ *security through obscurity*, in which the exact data contents are obfuscated to prevent significant analysis. For example, it is difficult for a non-insider to learn what information is contained in messages with arbitration ID 704, even after significant reverse engineering efforts [4], [5].

These obfuscation processes do not mask the vehicle itself: CAN messages from one vehicle are uniquely identifiable. As this research shows, an attacker can devise a system capable of distinguishing CAN messages from different vehicles. Such a

system does not require massive data troves, and such a system is not limited by the CAN's low computational power.

B. Deep Learning

In deep learning, convolutional neural networks (CNNs) "are a specialized kind of neural network for processing data that has a known grid-like topology"; thus, these networks are well-suited for time-series data segments, which are effectively one-dimensional grids [6]. By concatenating all sequential CAN messages with the same arbitration ID, we obtain some unknown number[1] of time series.

This research shows that a CNN can learn to identify which vehicle from a set of vehicles generated some given time series. The more interesting problem, however, is determining whether or not two time series come from the same vehicle. A deep learning model capable of distinguishing signals in this way is capable of learning learning to distinguish any number of cars, even if some of these cars are not in the training set.

For this, we use a siamese neural network. According to [7], "the Siamese network has two input fields to compare two patterns and one output whose state value corresponds to the similarity between the two patterns." Often, the *triplet loss* function computes this similarity measure; effectively, this function minimizes the distance between two examples from the same class and maximizes the distance between two examples from different classes [8]. As Figure 2 illustrates, the architectures of the two sides of an SNN are identical, and thus the training process adjusts the weights in both. Although most researchers apply SNNs to image recognition, [9] provides an example of determining sequence similarity with a siamese neural network.

C. Related Work

As discussed in [10], most researchers concerned with time series data attempt to predict future values, especially those researchers focused on CAN security. For example, [11] devises an intrusion detection system (IDS) by analyzing sequential CAN IDs, and [12] creates an IDS that learns relationships in different time series signals. However, some researchers in recent years have used deep learning to classify CAN data. For example, [13] train a deep neural network to classify CAN data packets as either *normal* or *abnormal*. The literature survey in [14] details more examples of deep learning as applied to IDSs and CAN security.

Recent research shows that one can certainly distinguish CAN data. The researchers in [15] demonstrate that machine learning can effectively discriminate between different drivers using just the CAN data collected from the vehicle. However, this research uses a set number of drivers, and it does not claim to distinguish time series generated by different vehicles.

It should now be clear why this research is necessary: various research efforts a) predict CAN data but do not classify it, b) learn a specific number of classes by analyzing reverse-engineered signals, or c) classify CAN signals as good or bad.

[1]It is difficult to determine exactly how many unique signals are described by a single arbitration ID.

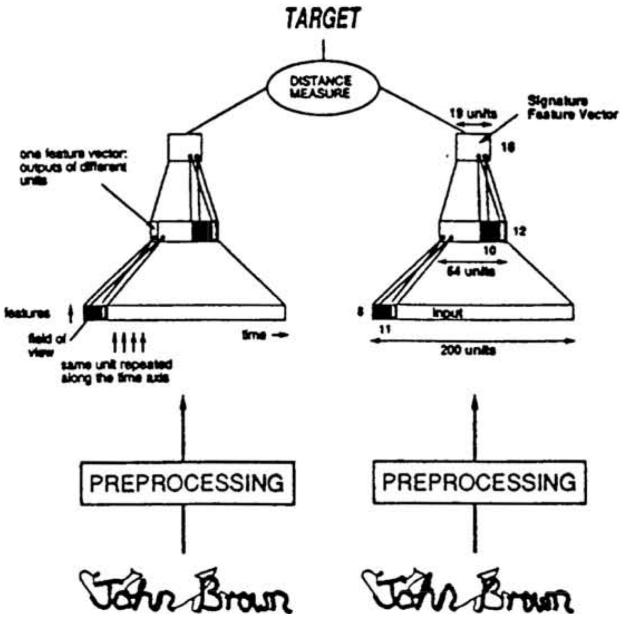

Fig. 2. A simple visual depiction of the structure of a siamese neural network. This network learns to distinguish signatures. The model outputs *same* or *not same* and serves as a signature verification system.

None of these efforts demonstrate an ability to *classify* an *unknown* number of vehicles using *raw* CAN data. This is an important gap in current research, and this research seeks to address this gap.

In the following section, we detail the experimental methodology, including the data's origins and structure and the data-formatting process. Additionally, we describe the deep learning models (i.e., their architectures and hyperparameters) and the model evaluation strategy.

III. Methodology

This research shows that CAN data uniquely identifies a vehicle. We train two different deep learning models—one a standard CNN and the other an SNN—to determine how well a model can distinguish between CAN data snapshots from different vehicles. Every snapshot in the dataset, which comes from ORNL and from Captain Stone's doctoral research, contains 1) 1,024 sequential bytes of CAN data and 2) a vehicle ID. Input to the models consist of the data bytes; the model then predicts the vehicle ID and compares its output to the true vehicle IDs. This is thus a supervised classification problem, and so it is likely that supervised techniques can learn to discriminate between snapshots from distinct vehicles.

The remainder of this section discusses the data in detail: its nature and origins, the cleanup and formatting process, and the classes and their distributions. Additionally, this section describes the deep learning models employed, including the architectures and the model-fitting and evaluation processes.

A. Data

This research uses data from two sources. Oak Ridge National Laboratory recently conducted at least 52 CAN data captures on nine different vehicles. Some of those 52 captures

TABLE I
MAKE, MODEL, AND YEAR FOR THE TWENTY VEHICLES

Vehicle	Make	Model	Year
1	Toyota	Tacoma	2008
2	Toyota	Corolla	2009
3	Nissan	Leaf	2011
4	Ford	C-Max	2013
5	Chevrolet	Volt	2015
6	Ford	F-150	2014
7	Ford	Fusion	2016
8	Subaru	WRX	2017
9	Subaru	Outback	2009
101	Chevrolet	Cobalt	2009
102	Chevrolet	Silverado	2011
103	Dodge	1500	2014
104	Ford	F-150	2017
105	Ford	Focus	2010
106	Honda	Accord	2012
107	Honda	Accord	2015
108	Nissan	370Z	2015
109	Nissan	XTERRA	2010
110	Saab	9-7X	2009
111	Toyota	Corolla	2009

TABLE II
NUMBER OF ARBITRATION IDS AND SNAPSHOTS PER VEHICLE

Vehicle	ArbIDs	Snapshots	Proportion
1	8	4440	1.49 %
2	47	6895	2.32 %
3	49	141841	47.72 %
4	103	14631	4.92 %
5	112	43377	14.59 %
6	79	9511	3.20 %
7	124	35142	11.82 %
8	35	5018	1.69 %
9	21	8211	2.76 %
101	29	4102	1.38 %
102	50	1824	0.61 %
103	62	1756	0.59 %
104	78	2182	0.73 %
105	24	3791	1.28 %
106	28	1695	0.57 %
107	42	2198	0.74 %
108	38	3020	1.02 %
109	26	2553	0.86 %
110	19	2974	1.00 %
111	38	2083	0.70 %

TABLE III
EXAMPLES OF DATA SNAPSHOTS

Vehicle	Capture	ArbID	Data
3	8	1532	074 166 111 254 255 240 254 ...
7	24	535	003 212 003 209 003 199 003 ...
108	108	292	255 248 000 128 015 254 030 ...
111	111	624	026 000 065 073 073 064 000 ...

contain sensitive information, so the lab shared the data from the 34 non-confidential captures. Additionally, recent Air Force Institute of Technology graduate Brent Stone conducted one capture on each of 11 different vehicles for his own doctoral dissertation. In total, this research uses 2.44 gigabytes of data from ORNL and another 230 megabytes of Captain Stone's CAN data. Table I displays metadata for the 20 vehicles in the dataset.

A large comma-separated values (CSV) file stores the data. Each of the 44,893,238 rows in this file represents one CAN message, and each row contains:

- a `capture_id` and a `vehicle_id`, which identify the message's origin,
- a `timestamp` relative to the start of the capture,
- an `arbitration_id`, which serves as the vehicle's identifier of the message's contents,
- a `dlc`, or *data length code*, which indicates how many bytes of data the message contains,
- the hexadecimal `data` itself,
- and the vehicle's `make`, `model`, and `year`.

The messages in this CSV file are not immediately suitable for deep learning. Each message contains no more than eight data bytes (and many contain fewer), and it is extremely unlikely that so little information can uniquely identify a vehicle. Additionally, disparate arbitration IDs typically contain disparate information. For example, arbitration ID 704 might detail a vehicle's instantaneous velocity while 1A3 contains the same vehicle's gear setting. A model that does not discriminate between the various IDs—that is, a model that trains on the messages as they appear in the CSV file—is not likely to learn the nuances of a given vehicle's CAN traffic.

To address this issue, we build a set of data snapshots, where each snapshot contains 1,024 bytes of sequential CAN data from one arbitration ID of one capture. Specifically, for each arbitration ID in each capture, we sort the messages by timestamp before splitting all hex data into a list of hex bytes, converting the hex bytes into integers, and dividing the list into snapshots of 1,024 integers. Because each integer represents one byte of CAN data, each snapshot contains $8*1024 = 8192$ data bits. Then, for arbitration IDs with at least one full snapshot[2], we write each snapshot and the associated vehicle ID to a new CSV file. This makes it more likely that a) each snapshot contains sufficient information and that b) the underlying CAN data structure is preserved.

This new CSV file contains 297,244 snapshots from 1,012 arbitration IDs of 20 vehicles. Table II illustrates the distributions of arbitration IDs and snapshots over all vehicles (the number of arbitration IDs for a given vehicle is innate; the number of snapshots, on the other hand, depends primarily on the length of the data capture). Clearly, Vehicles 3, 5, and 7 contain a large majority of all snapshots, and Vehicle 3 contains nearly half. As described in Section III-C, we address this class imbalance during model training.

Table III presents a few example snapshots. The deep learning model receives these snapshots as input and attempts to predict the vehicle that generated each snapshot; the model does *not* receive the capture or arbitration ID. This ensures the model cannot simply learn which captures/arbitration IDs map to each vehicle.

B. Model Architecture

To demonstrate that a given vehicle's CAN data uniquely identifies said vehicle, we train and compare deep learning models on multiple data snapshots. As described in Section

[2]Arbitration IDs without at least one full snapshot are not likely to positively contribute to the model, so these IDs are ignored.

III-A, each snapshot contains sequential CAN data from one arbitration ID of one data capture, and each is labeled with one of 20 vehicles, so this is a supervised classification problem.

A standard deep learning classifier, like a vanilla CNN, serves as a proof of concept. Such a classifier cannot easily generalize to new data, but it can learn to distinguish between all vehicles in the training set. In other words, standard CNNs learn the distinguishing features for each class, so these networks cannot properly identify new vehicles, but a well-trained CNN can effectively determine a sample's class if it was trained on the class.

A siamese neural network, on the other hand, is extensible to new data, and thus SNNs are the focal point of this research.[3] SNNs learn why samples from two classes are different and why samples from one class are the same. The difference is subtle yet crucial. An SNN is more likely to succeed on a new vehicle X because it does not care about Vehicle X's specific data; it only cares about whether this data is the same as or different from the data for some other vehicle.

Because each snapshot contains sequential, one-dimensional data, and because the task at hand is classification, both models utilize one-dimensional convolutional layers. In testing, the standard CNN receives one snapshot as input and outputs one of the 20 classes. The SNN receives two snapshots as input and outputs either *same class* or *different classes*. Sections III-B1 and III-B2 describe the networks in detail.

1) The Standard Deep Learning Model: The standard model utilizes a typical CNN structure consisting of convolutional, pooling, and dropout layers. Figure 3 displays the specific architecture, including the layers and input/output shapes. The convolutional layers use a kernel size of three, the pooling layer pools over two elements at each step, and the dropout layer sets 50% of the inputs to zero. The convolutional layers and the first dense layer all use a `relu` activation function; the final dense layer uses `softmax`. In total, the model has 3,278,984 parameters, and all but 128 are trainable.

[This is still a very early model. I need to tune it/conduct a hyperparameter search. As such, my model parametrization decisions are more like parametrization initial-guesses.]

2) The Siamese Neural Network: [I still need to find more siamese neural network resources before I can build my own. For this reason, I don't yet have an architecture.]

C. Model Fitting

In the full snapshot dataset, the classes are significantly imbalanced. For example, nearly 48% of all snapshots come from Vehicle 3; one should expect each vehicle to contain about 5% of the snapshots. Vehicle 106 contains the fewest snapshots at 1,695, or 0.57% of the full dataset. We could randomly sample $n = 1695$ snapshots from every vehicle to ensure equal representation for all classes, but model performance is likely greater if we use all available training data by employing tools meant for classifying imbalanced datasets. One such tool is scikit-learn's `class_weight` module, which adjusts weights during training to account for the frequency of each class.

[3]An SNN, at its core, consists of two neural networks; in this research, the networks that form the SNN are CNNs.

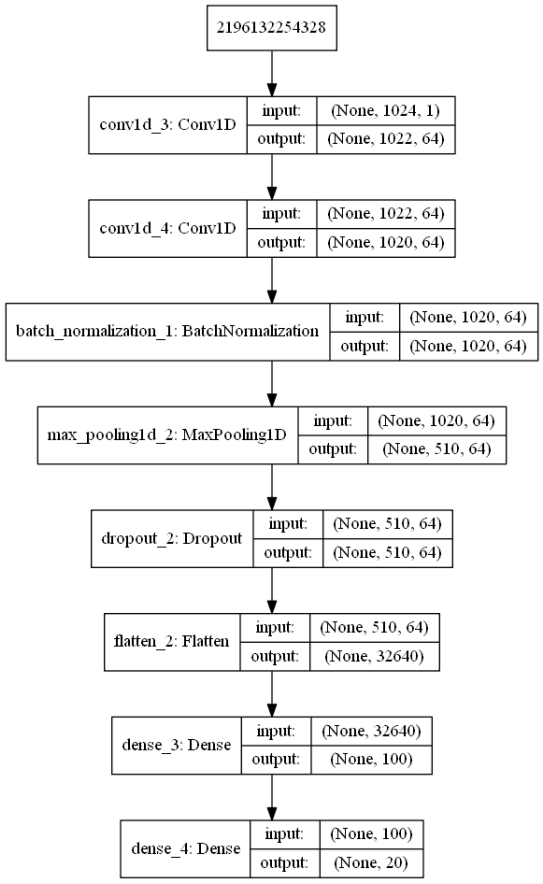

Fig. 3. The layers of the standard convolutional neural network

This module ensures that both models can use all training data without unnecessarily suffering from class imbalance.

To properly train and evaluate the models, we split the dataset into three sets: 60% of the snapshots are used in training, 20% in validation, and 20% in testing. Scikit-learn's `train_test_split` enables this split by randomly sampling from the full set of snapshots.

Another important tool during model training is Keras's `EarlyStopping` callback, which limits overfitting. This function monitors validation loss during training and terminates the training if said loss does not improve in 10 epochs. The callback then restores the model to its best possible version. This tool also limits model training time.

However, these tools do not guarantee optimality, so we also iteratively improve the models through repeated testing and hyperparameter tuning. A search over a set of possible hyperparameter configurations ensures we can identify the best classifier. *[I do not have time to conduct this hyperparameter search before the draft is due. I intend to do so after I submit Homework 4.]*

D. Model Evaluation & Analysis

To effectively quantify model performance, we compute classification metrics for the two models on the full dataset and on various subsets of the dataset, including:

- Captures from ORNL;

- Captures from Captain Stone's research;
- Captures from specific automakers, including Chevrolet, Subaru, and Nissan;
- Captures from specific models, like the Ford F-150, Honda Accord, and Toyota Corolla; and
- A fully balanced set of snapshots (that is, a set containing $n = 1695$ random snapshots from each class).

The classification metrics include accuracy, recall, and F-score. In classification, different tasks require different performance measures, so utilizing these various metrics should encompass a slew of possible CAN data classification problem domains.

We also investigate the models' misclassifications to identify possible reasons for the errors. For example, Manufacturer X could use the exact same CAN configuration for different models, so even a well-trained model could fail to distinguish between Manufacturer X's vehicles. The results of these investigations guide further model modifications and tuning.

The remainder of this report details the results of the experiment. Specifically, Section IV presents and discusses the results and implied performance of the two models. Section V describes the implications of these results and suggests a few areas for future research.

IV. Results & Discussion

Early results are promising. The standard CNN achieves a test-set accuracy of 93.21% after just 16 training epochs. This is certainly a strong result, but this research requires further analysis into what exactly the model misclassifies. Additionally, we must tune the initial CNN so as to identify the optimal model, and we must develop and evaluate an SNN before we can compare the performance of the two models.

Identifying why the standard CNN misclassifies snapshots may provide insight into the inherent structure of CAN data, but such a model is not likely useful in real-world problems. However, because a well-trained SNN is extensible and can easily learn to distinguish new vehicles—even with few data—strong results from our SNN will prove noteworthy. Specifically, strong results indicate that the auto manufacturer's policy of security through obscurity is insufficient in ways not previously analyzed. Additionally, strong results imply that deep learning is another useful tool for CAN data analysis; this may assist future researchers in their vehicle security research.

[My actual results will go here (e.g., precision-recall curves for each class in each model, a table showing the accuracy, recall, and F-score for each data subset for each model).]

The final section discusses the real-world implications of these results. Additionally, the section offers areas for future research in CAN analysis and security.

V. Conclusions & Future Work

References

[1] National Instruments, "Controller area network (CAN) overview." https://www.ni.com/en-us/innovations/white-papers/06/controller-area-network–can–overview.html, 2019. [Online; accessed 8-July-2019].

[2] R. B. GmbH, "CAN specification version 2.0," 1991.

[3] S. C. HPL, "Introduction to the controller area network (CAN)," *Application Report SLOA101*, pp. 1–17, 2002.

[4] R. Buttigieg, M. Farrugia, and C. Meli, "Security issues in controller area networks in automobiles," in *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 93–98, IEEE, 2017.

[5] B. Stone, S. Graham, B. Mullins, and C. Schubert Kabban, *Enabling Auditing and Intrusion Detection for Proprietary Controller Area Networks*. Dissertation, Air Force Institute of Technology, 2018.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[7] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Advances in neural information processing systems*, pp. 737–744, 1994.

[8] A. Burkov, *The Hundred-Page Machine Learning Book*. 2019.

[9] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[10] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Upper Saddle River, NJ: Prentice-Hall, Inc., 1994.

[11] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," pp. 1577–1583, 06 2017.

[12] Z. Tyree, R. A. Bridges, F. L. Combs, and M. R. Moore, "Exploiting the shape of CAN data for in-vehicle intrusion detection," *CoRR*, vol. abs/1808.10840, 2018.

[13] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," pp. 1–5, 05 2016.

[14] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, 09 2017.

[15] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 34–50, 2016.