Fingerprinting Vehicles With CAN Bus Data Snapshots

David R. Crow, Scott R. Graham, Brett J. Borghetti
Air Force Institute of Technology, Wright-Patterson Air Force Base, USA
david.crow@afit.edu
scott.graham@afit.edu
brett.borghetti@afit.edu

Abstract: Today's vehicle manufacturers do not tend to publish proprietary controller area network (CAN) packet formats. This is a form of *security through obscurity* – it makes reverse engineering efforts more difficult for would-be intruders – but obfuscating the CAN data in this way does not adequately hide the vehicle's unique signature. Specifically, modern methods can identify a vehicle's signature in a segment of its CAN data, even if these data are unprocessed or limited in scope. In deep learning, this is a multiclass classification problem which asks the following question: given a sample of CAN data, can we determine which vehicle generated the sample? To answer this question, we employ two datasets, one from Oak Ridge National Laboratory (ORNL) and one from Stone et al (2018). ORNL's corpus is comprised of nearly 2.5 gigabytes of data captured on the CAN buses of nine different vehicles; Stone et al's (2018) corpus contains over 230 megabytes of data from 11 different vehicles. In this research, 1,024 bytes of sequential CAN data constitute one data sample, so formatting and partitioning the datasets gives almost three hundred thousand individual samples. We label every sample with its generating vehicle to enable fully supervised learning. We then train two distinct deep learning models on the data. The results indicate that a standard multilayer perceptron (MLP) can effectively classify these CAN data samples. The results also indicate that a deep convolutional neural network (CNN) can classify the samples at a greater performance level than can the MLP, but both models still surpass a balanced classification accuracy of 80% on the full dataset. Clearly, one can determine which vehicle generated a given sample of CAN data. This erodes consumer safety: a sophisticated attacker who establishes a presence on an unknown vehicle can use similar techniques to identify the vehicle and better format attacks.

Keywords: classification, deep learning, controller area network, CAN security, fingerprinting vehicles

1. Introduction

The CAN bus, which connects a large number of the devices in a modern vehicle (including vital systems like the brakes, steering wheel, and transmission), is vulnerable. It is vulnerable to cyberattacks capable of altering, preventing, or otherwise modifying the operator's desired behavior. This research presents a new vulnerability: the packets that broadcast on a vehicle's CAN bus uniquely identify the vehicle. This means that an attacker can construct a database of known CAN packet formats and, using this database and one of the tools presented here, identify a *new* vehicle. This allows the attacker to strengthen the attack and thus places the operator and passengers at greater risk.

We illustrate this vulnerability by building, tuning, and evaluating two deep learning models. When given a sample of CAN data from a specific vehicle, these models predict the generating vehicle 80% of the time. We train the models by giving as input CAN data samples and each sample's generating vehicle. We test these models by giving new samples as input and comparing the predicted vehicle to the actual vehicle. This research seeks to determine which of 20 different vehicles generated each test sample.

Data comes from two primary sources and consists of a two and a half gigabytes of raw CAN data captured from 20 distinct vehicles. By parsing, formatting, and partitioning the available data, we generate two disparate datasets. The first is composed of all available 1,024-byte CAN samples; the second contains samples from the first such that the classes are evenly represented. Each sample's target label is the ID of the vehicle that generated it.

We then feed these samples into the deep learning models. The first, an MLP, is simple enough that an attacker only somewhat familiar with deep learning tools can easily implement it and use it to classify CAN samples. The second, a more complex CNN, achieves better classification performance than does the MLP.

Results indicate that one can use these deep learning methods to determine which vehicle generated a given CAN data sample. This risks vehicle operator and passenger safety because it gives bad cyber actors another tool to correctly structure malicious CAN packets.

The remainder of this report presents the research in detail. Section 2 examines some of the related work in current literature and explains why this work is insufficient for the research at hand. Section 3 describes the data, the deep learning model, the model-fitting process, and the model analysis and evaluation tools. Section 4 presents and discusses the results obtained. Section 5 explains the implication of these results and suggests possible opportunities for future research.

2. Background

To better understand this research, one must understand basic information about CANs, about deep learning, and about related research. Section 2.1 discusses the CAN, including its message structure and contents. Section 2.2 discusses time series classification approaches in deep learning and how they relate to this research. Section 2.3 examines related work in CAN analysis and fingerprinting vehicles, both with and without deep learning.

2.1 The controller area network

The CAN 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 (National Instruments, 2019).

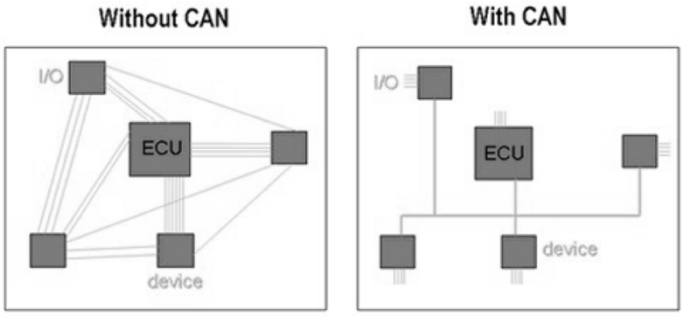

Figure 1: Vehicle network wiring with and without CAN

CAN messages are limited in capacity. Each one contains message overhead and up to 64 bits of data (Robert Bosch GmbH, 1991; Corrigan and Texas Instruments, 2008). Additionally, the components along the CAN do not possess a wealth of computing power, so computational capabilities are 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 (Buttigieg, Farrugia, and Meli, 2017; Stone et al, 2018).

These obfuscation processes do not mask the vehicle itself because 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 capability.

2.2 Deep learning

The MLP is a feedforward neural network, which means that information flows from the input layer, through each hidden layer, and to the output layer (Goodfellow, Bengio, and Courville, 2016). MLPs are often called vanilla neural networks because they are simple, no-frills networks; one can use an MLP to approximate a nonlinear function using few hidden layers (Burkov, 2019).

In deep learning, 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 (Goodfellow, Bengio, and Courville, 2016; Box, Jenkins, and Reinsel, 1994). By concatenating all sequential CAN messages with the same arbitration ID, we obtain multiple time series. We then tune a CNN to demonstrate its superior performance over the MLP.

2.3 Related work

As discussed in Box, Jenkins, and Reinsel (1994), most researchers concerned with time series data attempt to predict future values, especially those researchers focused on CAN security. For example, Marchetti and Stabili (2017) devise an intrusion detection system (IDS) by analyzing sequential CAN IDs, and Tyree et al (2018) create 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, Kang and Kang (2016) train a deep neural network to classify CAN data packets as either *normal* or *abnormal*. The literature survey in Kwon et al (2017) details more examples of deep learning as applied to IDSs and CAN security.

Recent research shows that one can certainly distinguish CAN data. Enev et al (2016), for example, demonstrate that machine learning can effectively discriminate between different drivers using just the reverse-engineered CAN data collected from the vehicle. However, this research uses a set number of drivers, and it does not claim to distinguish raw time series generated by different vehicles.

Clearly, various research efforts a) predict CAN data but do not classify it, b) classify vehicles by analyzing reverse-engineered signals, or c) classify CAN signals as good or bad. None of these efforts demonstrate an ability to classify vehicles using *raw* CAN data. This is an important gap in current research, and this research seeks to address this gap.

In Section 3, 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.

3. Methodology

We train two different deep learning models – one an MLP and the other a CNN – to determine how well a model can distinguish between CAN data samples from different vehicles. Every sample in the dataset, which comes from ORNL and from Stone et al (2018), contains a vehicle ID and 1,024 sequential bytes of CAN data. Input to the models consists of the data bytes; the model then predicts the vehicle ID and compares its output to the true vehicle ID.

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.

3.1 Data

This research uses data from two sources. Oak Ridge National Laboratory recently conducted 34 CAN data captures on nine different vehicles. Stone et al (2018) conducted one capture on each of 11 different vehicles. In total, this research uses 2.44 gigabytes of data from ORNL and another 230 megabytes of Stone et al's (2018) CAN data. Table 1 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.

Table 1: 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

To address this issue, we build a set of data samples, where each sample 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 samples of 1,024 integers. Because each integer represents one byte of CAN data, each sample contains $8 \times 1024 = 8192$ data bits. Then, for arbitration IDs with at least one full sample, we write each sample and the associated vehicle ID to a new CSV file. This makes it more likely that each sample contains sufficient information and that the underlying CAN data structure is preserved.

Table 2 presents a few example data samples. The deep learning model receives these samples as input and attempts to predict the vehicle that generated each sample; 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.

Table 2: Examples of CAN data samples

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 …

The new CSV file contains 297,244 samples from 1,012 arbitration IDs of 20 vehicles. Table 3 illustrates the distributions of arbitration IDs and samples over all vehicles (the number of arbitration IDs for a given vehicle is innate; the number of samples, on the other hand, depends primarily on the length of the data capture). Vehicles 3, 5, and 7 contain a large majority of all samples. Vehicle 3 alone contains nearly half. As described in Section 3.3, we address this class imbalance during model training.

Table 3: Number of arbitration IDs and samples per vehicle

Vehicle	ArbIDs	Samples	Proportion
1	8	4440	1.49 %
2	47	6895	2.32 %
3	49	141841	47.72 %

Vehicle	ArbIDs	Samples	Proportion
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 %

3.2 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 samples. As described in Section 3.1, each sample contains sequential CAN data from one arbitration ID of one data capture, and each is labeled with one of 20 vehicles.

Using an MLP to classify vehicles is a naive approach, but it is useful to present the naive approach's performance to demonstrate that an attacker does not need complex methods to adequately classify vehicles. Each CAN sample contains 1,024 data bytes, so the MLP receives 8,192 bits as input and outputs one of the 20 classes. Section 3.2.1 describes the MLP in detail.

We also present a more complex approach – a CNN – to demonstrate that a sophisticated attacker can achieve increased performance over the naive approach. The CNN utilizes one-dimensional convolutional layers because each sample contains sequential, one-dimensional data. In testing, the CNN receives one sample as input and outputs one of the 20 classes. Section 3.2.2 presents the details of the model.

3.2.1 The multilayer perceptron

We test multiple hyperparameter configurations to identify the best MLP; the left panel of Figure 2 displays this model. It has three hidden layers, each of which has 512 nodes and uses a `relu` activation function. The output layer contains 20 nodes – one for each vehicle in the dataset – and uses a `softmax` activation function. We compile the model with an Adam optimizer of learning rate $lr = 0.001$ and a categorical cross-entropy loss function. The model has 1,060,372 parameters, all of which are trainable. Every MLP implementation utilizes this basic architecture. In total, we test 27 MLP architectures; Table 4 lists the various options for each hyperparameter used during model construction.

Table 4: Hyperparameter options for MLP construction

Hyperparameter	Options
Number of hidden layers	1, 2, 3
Number of nodes per hidden layer	128, 256, 512
Optimizer learning rate	0.01, 0.001, 0.0001

3.2.2 The convolutional neural network

Like with the MLP, we test multiple hyperparameter configurations before selecting the best CNN. The best model utilizes a typical CNN structure consisting of convolutional, pooling, and dropout layers. The right panel of Figure 2 displays the specific architecture. The convolutional layers each use 64 filters with a kernel size of two, the pooling layer pools over three elements at each step, and the dropout layer sets 50% of its inputs to zero. The convolutional layers and the first dense layer all use a `relu` activation function; the final dense layer

uses `softmax`. We again compile the model with an Adam optimizer of learning rate $lr = 0.001$ and a categorical cross-entropy loss function. In total, the model has 2,186,824 parameters, and all but 128 are trainable. Every CNN implementation utilizes this basic architecture. In total, we test 18 CNN architectures; Table 5 lists the various options for each hyperparameter used during model construction.

Table 5: Hyperparameter options for CNN construction

Hyperparameter	Options
Filters per convolutional layer	32, 64
Kernel size per filter	2, 3, 4
Pooling size	2, 3, 4

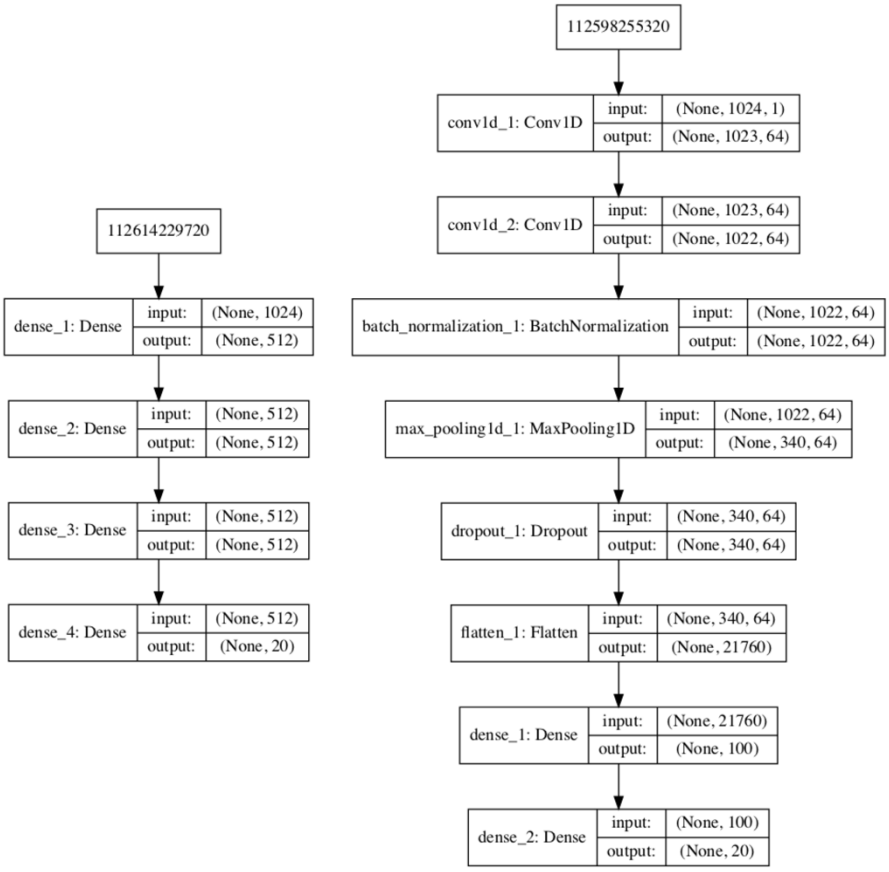

Figure 2: The best architecture identified in an iterative model evaluation process. *Left:* MLP; *Right:* CNN.

3.3 Model fitting

Table 3 shows that the classes in the full dataset are significantly imbalanced. For example, nearly 48% of all samples come from Vehicle 3; one should expect each vehicle to contain about 5% of the samples. To address this imbalance, we use scikit-learn's `class_weight` module during model training; this tool adjusts weights to account for the frequency of each class, and thus it ensures that both models can use all training data without unnecessarily suffering from class imbalance. Additionally, we create a second, balanced dataset by randomly sampling $n = 1695$ samples from every vehicle.

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 the loss does not improve in 10 epochs. The callback then restores the model to its best version. This tool also limits model training time.

To properly train and evaluate the models, we split each dataset into three sets: 60% of the samples 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 dataset. We train each model on the training set, terminate training with the early stopping callback (which monitors the validation set's loss), and evaluate the trained model on the testing set. 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. In total, we evaluate 27 different MLPs and 18 different CNNs. Tables 4 and 5 depict the hyperparameter options for these models.

Figure 3 shows that the final models possess sufficient capacity for this task. Specifically, the validation loss and accuracy for each model is nearly as good as the training loss and accuracy, and all four losses and accuracies appear to reach an asymptote. This means that additional training will not benefit the models, and it also means that the models are nearly optimal for this task.

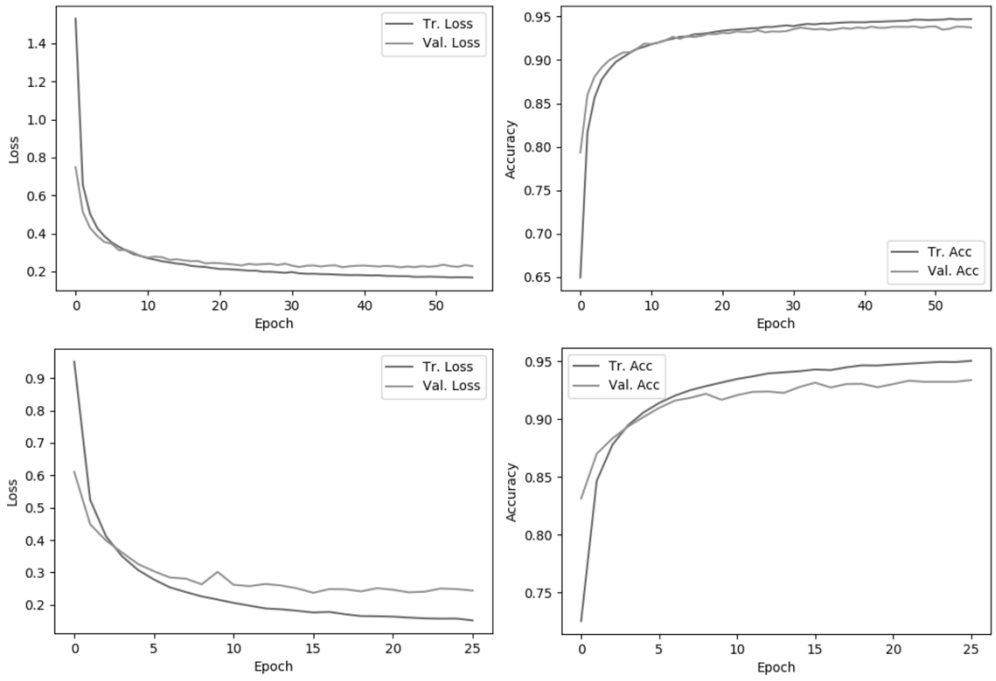

Figure 3: Training and validation losses and accuracies over time for the two final models. *Top left:* loss over time for the MLP; *Top right:* accuracy over time for the MLP; *Bottom left:* loss over time for the CNN; *Bottom right:* accuracy over time for the CNN.

3.4 Model evaluation and analysis

We present results for the MLP to demonstrate that an intruder with a low-level understanding of deep learning can use this field to better format CAN attacks. We compare the MLP's performance to that of a CNN to show that a sophisticated intruder can use more advanced techniques and achieve better results.

To evaluate model performance, we compute balanced accuracy for both the balanced and the imbalanced dataset. This metric takes into consideration the various class distributions when computing the classification accuracy. Even for the balanced dataset, this is important. Because we randomly sample from the dataset to build the training, validation, and testing sets, each of those three sets could be imbalanced.

Additionally, we present confusion matrices to allow for an in-depth analysis of where each model fails. 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. These analyses may even imply a similar CAN structure across different makes, so the results of these investigations guide further model modifications and tuning.

The remainder of this report details the results of the experiment. Specifically, Section 4 presents and discusses the results and implied performance of the two models. Section 5 describes the implications of these results and suggests a few areas for future research.

4. Results

This section presents the classification results for the two model types on the two datasets. Results indicate that both the MLP and the CNN can adequately classify CAN data segments, but the CNN is certainly better than the MLP on both the imbalanced dataset and the balanced dataset. Results also indicate that both models perform better on the larger, imbalanced dataset; this is simply due to the availability of training data in each set. Table 6 presents the overall balanced accuracies.

Table 6: Balanced accuracy for each model on each dataset

Model	Dataset	Accuracy
Multilayer perceptron	Imbalanced	81.71 %
Multilayer perceptron	Balanced	76.95 %
Convolutional neural network	Imbalanced	83.18 %
Convolutional neural network	Balanced	80.94 %

Table 7 presents class-specific accuracy values for the four trained models. The grey cells indicate the best model/dataset combination for each of the 20 vehicles. Clearly, the CNN trained on the imbalanced dataset performs better than the other models on a majority (13/20) of the classes, and it does not give the worst overall performance for any of the classes. This is further evidence of the superior performance of the CNN, and it also indicates that more training data is better, even if the classes in the data are severely imbalanced.

Table 7: Class accuracy for each model on each dataset

Vehicle	MLP, Imbalanced	MLP, Balanced	CNN, Imbalanced	CNN, Balanced
1	98.31 %	83.29 %	92.58 %	88.20 %
2	86.89 %	76.61 %	88.69 %	80.88 %
3	93.59 %	81.82 %	93.60 %	74.21 %
4	94.40 %	76.52 %	94.55 %	77.87 %
5	96.85 %	70.72 %	97.56 %	81.57 %
6	96.20 %	94.72 %	94.72 %	83.63 %
7	89.50 %	70.47 %	92.71 %	48.27 %
8	94.99 %	92.49 %	96.85 %	89.34 %
9	96.03 %	86.26 %	98.34 %	92.64 %
101	87.54 %	60.22 %	89.00 %	83.38 %
102	97.63 %	84.31 %	82.51 %	75.56 %
103	85.67 %	89.51 %	93.57 %	81.59 %
104	83.18 %	89.55 %	89.29 %	87.83 %
105	91.80 %	50.83 %	93.94 %	94.83 %
106	89.38 %	89.70 %	85.59 %	85.17 %
107	82.52 %	69.44 %	87.41 %	79.26 %
108	94.42 %	73.61 %	95.12 %	89.64 %
109	90.79 %	85.71 %	93.60 %	82.67 %
110	82.18 %	84.50 %	83.78 %	87.81 %
111	74.01 %	70.98 %	88.07 %	78.80 %

Figure 4 presents the multiclass confusion matrices for both model types on both datasets. Generally, one can see that the models tend to correctly classify CAN segments, but some specific misclassifications are certainly common. For example, all four models trained models misclassify Vehicle 106 as Vehicle 107 (and vice versa) to some extent; these are different-year Honda Accords. Similarly, the models often confuse Vehicles 2 and 111, the two 2009 Toyota Corollas in the dataset. One can also see that the models usually perform similarly on the same vehicle. For example, all models demonstrate strong performance on Vehicle 8 – perhaps Subaru employed a distinct CAN format when designing its 2017 WRX. Conversely, none of the models achieve incredibly strong performance on Vehicle 111. One can easily discern other trends by further analyzing Figure 4.

Section 5 discusses the real-world implications of these results. Additionally, the section offers areas for future research in CAN analysis and security.

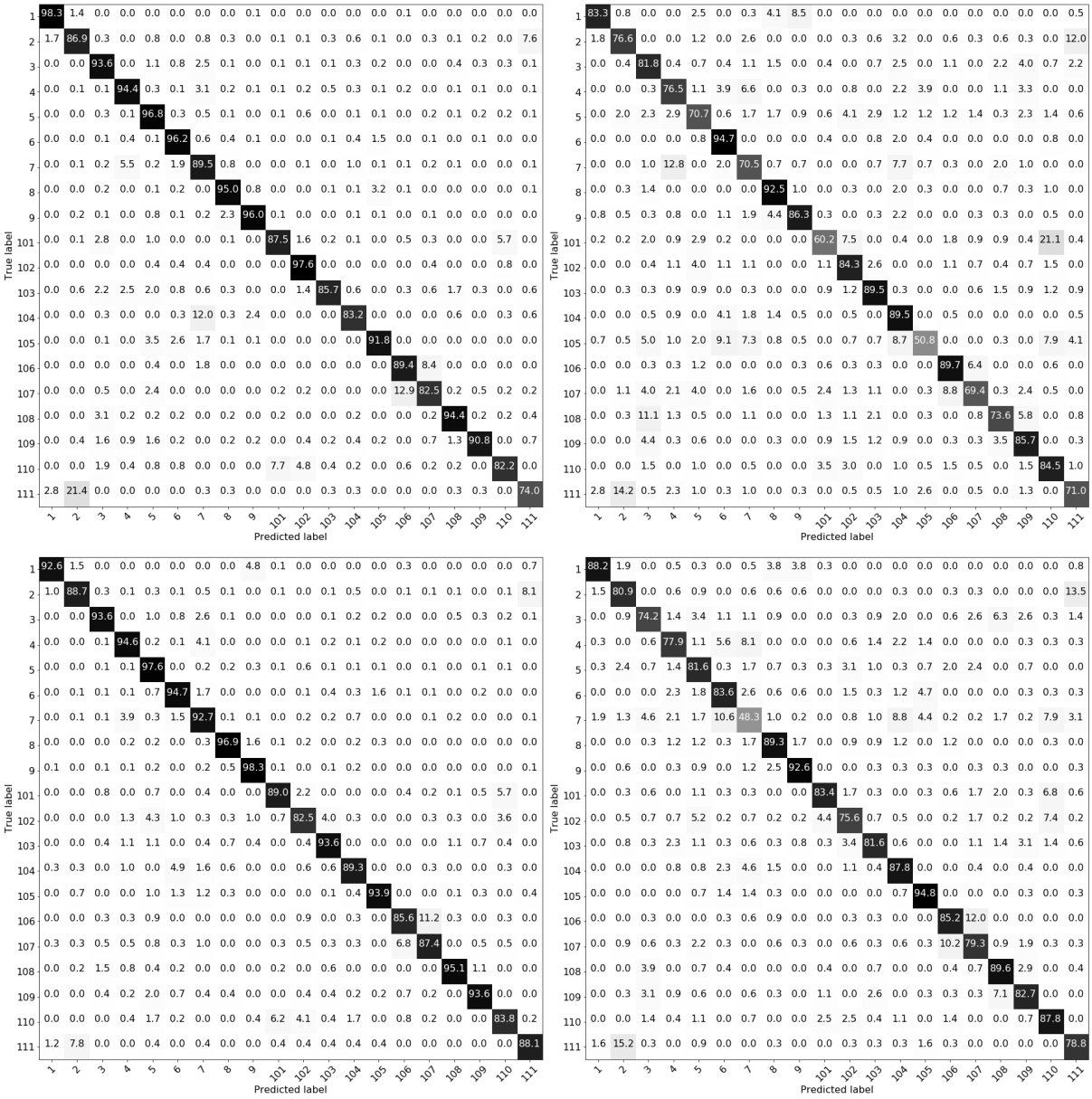

Figure 4: Confusion matrices. *Top left:* MLP on the imbalanced dataset; *Top right:* MLP on the balanced dataset; *Bottom left:* CNN on the imbalanced dataset; *Bottom right:* CNN on the balanced dataset.

5. Conclusion

This research has two primary conclusions:

- A vehicle's CAN data uniquely identifies said vehicle; and
- An attacker can use one of the tools presented to improve attacks on vehicle-based CANs.

Future work should examine and improve upon the consistency of the data used in this research, and future work should also evaluate other, more robust methods for vehicle fingerprinting.

Clearly, one can fingerprint vehicles with CAN data samples. The best classifier in our research uses a CNN trained on an imbalanced dataset, and it achieves a balanced classification accuracy of 83.18%. Further model tuning could potentially yield a stronger model capable of better distinguishing CAN samples. One can even utilize more sophisticated classification techniques if one so desires. Still, our best model's performance is significant because it indicates that deep learning methods can fingerprint a vehicle using only its CAN data.

This means a malicious intruder can employ deep learning for personal gain. Such an attacker can use a well-tuned CNN (or an MLP if resources or knowledge are limited) to build a database of known vehicles; the attacker can then reference this database when intruding on a new CAN bus to better structure the desired attack. This presents a risk to today's vehicles.

Devising a siamese neural network (SNN) capable of learning the difference between CAN segments is the logical next step for this research. An SNN can generalize to new classes because it learns why two observations come from the same class or from different classes. Standard classifiers (e.g., MLPs and CNNs), on the other hand, must learn every class. For this reason, an SNN is extensible: one can train an SNN on a set of vehicles and, because it knows why samples from two vehicles are different, one can introduce many new vehicles and still maintain solid performance. This is a boon for the malicious intruder: such an attacker need not collect CAN segments from every potential vehicle to effectively attack new vehicles.

One other avenue for future work concerns CAN data consistency. We cannot guarantee that all data captures used the same route, driving conditions, or even driver, so it is possible that the deep learning models learned to classify these characteristics instead of the CAN packet structures. (Stone et al (2018) captured data along the same route and in the same conditions for all vehicles, so the strong classification performance of Vehicles 106 and 107 – both Honda Accords – indicates that the models do learn CAN structure to some extent.) For this reason, future work could entail a similar experiment on a more consistent corpus, one in which the researchers hold constant more of the variables during data collection.

Overall, our findings indicate a significant CAN vulnerability that an attacker familiar with deep learning can easily exploit. A bad actor who gains access to an unknown vehicle's CAN bus can employ deep learning to identify the vehicle – or at least determine that the vehicle is not one of the known vehicles – and improve the desired attack. This risks operator and passenger safety, and it warrants action on the part of automobile manufacturers. It also warrants further research into other deep learning applications on the CAN.

References

Box, G.E., Jenkins, G.M., and Reinsel, G.C. (1994) *Time Series Analysis: Forecasting and Control*. Prentice-Hall, Inc., Upper Saddle River, New Jersey.

Burkov, A., (2019) *The Hundred-Page Machine Learning Book*, Andriy Burkov.

Buttigieg, R., Farrugia, M. and Meli, C. (2017) "Security Issues in Controller Area Networks in Automobiles", *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, pp 93-98.

Corrigan, S. and Texas Instruments, (2008) "Introduction to the Controller Area Network (CAN)", *SLOA101A*, Dallas, TX.

Enev, M., Takakuwa, A., Koscher, K. and Kohno, T. (2016) "Automobile Driver Fingerprinting", *Proceedings on Privacy Enhancing Technologies*, Vol. 2016, No. 1, pp 34-50.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*, The MIT Press, Cambridge, Massachussetts.

Kang, M.J. and Kang, J.W. (2016) "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security", *PLOS ONE*, Vol. 11, No. 6, pp 1-17.

Kwon, D., Kim, H., Kim, J., Suh, S.C., Kim, I. and Kim, K.J. (2017) "A Survey of Deep Learning-Based Network Anomaly Detection", *Cluster Computing*, Vol. 22, No. 1, pp 949-961.

Marchetti, M. and Stabili, D. (2017) "Anomaly Detection of CAN Bus Messages Through Analysis of ID Sequences", *2017 IEEE Intelligent Vehicles Symposium*, pp 1577-1583.

National Instruments (2019) *Controller Area Network (CAN) Overview*, [online], National Instruments, https://www.ni.com/en-us/innovations/white-papers/06/controller-area-network--can--overview.html.

Robert Bosch GmbH, (1991) *CAN Specification Version 2.0*, Robert Bosch GmbH, Stuttgart, Germany.

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

Tyree, Z., Bridges, R.A., Combs, F.L. and Moore, M.R. (2018) "Exploiting the Shape of CAN Data for In-Vehicle Intrusion Detection", *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp 1-5.