Pattern Recognition and Machine Learning

Nils Weiss

March 17, 2020

1 Problem Definition

The major communication technology for real time data in modern cars are Controller Area Networks (CANs). CAN is a message-based communication protocol with an identifier field and eight bytes of payload. During normal operation of a car, the identifier field of a CAN message is used to specify the type of data. The eight bytes data field contain various real time values. Every message is broadcast on a shared bus. An Electronic Control Unit (ECU) can filter on the message identifier to evaluate if a message is relevant for its internal functionality. To define the data which is transported over CAN, Original Equipment Manufacturers (OEMs) are using Data Base CAN (DBC) files or similar description formats for their internal development. These DBC files are proprietary and usually not available in the public. Currently, there is no efficient way, to reverse engineer these data description through the analysis of CAN communication data. The final question is:

Which techniques of pattern recognition and machine learning can be used to classify CAN communication data automatically?

1.1 CAN Data

An exchange of real time values between ECUs is done through CAN messages. During a black box security investigation of a modern car, this traffic can be recorded with various tools. A well known open source tool to record CAN communication data is `candump`[1]. A `candump` log entry consists of a timestamp, an interface description and the CAN frame with identifier and payload. `candump` produces the following output:

Listing 1: `candump` logfile format output

```
(1526379707.348777) can0 22A#8D8802
(1526379707.348778) can0 0C9#80329B0762001038
(1526379707.348778) can0 0D3#2C889B0762001038
```

[1] `https://github.com/linux-can/can-utils.git`

```
(1526379707.348780) can0 0F9#00C1000000000000
(1526379707.349777) can0 0F1#280200400000
(1526379707.349778) can0 1A1#002041407D616200
(1526379707.349778) can0 1A3#8000000000000000
(1526379707.350744) can0 1C3#07530764FF106203
(1526379707.350745) can0 1C4#0062C7050062019E
(1526379707.350746) can0 1C5#2C952BEF2CE1019E
(1526379707.350747) can0 1DF#8518000000000000
(1526379707.350747) can0 1F4#0080000000000000
(1526379707.352205) can0 1F5#E500000F18000100
(1526379707.352207) can0 287#C10000
(1526379707.352211) can0 451#000000008400
(1526379707.354353) can0 1E5#5CFFBF3000000286
(1526379707.354355) can0 530#00000000
(1526379707.355971) can0 0C1#B34E6313321583F3
(1526379707.355973) can0 0C5#F1CD73D3B1D5F242
(1526379707.355975) can0 0D1#C000FFFD00FD00
(1526379707.355975) can0 185#1808
```

This log-file stores one CAN message per line. A line contains a timestamp, the interface name and the CAN message with identifier and data, both in hexadecimal representation. The number sign (#) is used to separate the identifier from data.

1.2 DBC file format

The DBC file format is described by this website: `http://socialledge.com/sjsu/index.php/DBC_Format`. The main goal of this research is the classification of all described data-types.

2 Feature Extraction

The CAN protocol and the DBC specification allow various early stage filtering of the log data.

All messages can be grouped by the CAN identifier since the identifier defines the data format transferred in the CAN frames data payload. Further group specific features can be extracted. All further explanations are only applied on CAN frames grouped by their identifier. Useful features are:

- Total number of messages per identifier
- Total number of flips per bit of two consecutive CAN frames
- Number of different values of a data field
- Frequency of a CAN message with a specific identifier

- Variance of the difference between two consecutive CAN frames with the same identifier

3 Data-field recognition

Knowledge about communication data is crucial for an automotive security researcher. Attacks or penetration tests which target the CAN bus communication require deep knowledge about the data format used for vehicle internal communication.

3.1 Transition Aggregation Vector (TAV)

A major feature for later analysis of CAN data is the TAV of a group of CAN frames. Let's define the data bits of a CAN frame as vector $x := (b_{63}, b_{62}, \ldots, b_0) \in \mathbb{F}_2^{64}$. A group of data bits from the same CAN identifier can be represented as matrix

$$X = \begin{pmatrix} b_{63,0} & b_{62,0} & \cdots & b_{0,0} \\ b_{63,1} & b_{62,1} & \cdots & b_{0,1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{63,n} & b_{62,n} & \cdots & b_{0,n} \end{pmatrix} \in \mathbb{F}_2^{64 \times n}$$

This Matrix X represents all data frames of a entire log file. The TAV of all data bits is computed through this equation

$$y_j = \sum_{i=1}^{n} (X_{j,i-1} \oplus X_{j,i}) \qquad y \in \mathbb{N}_0^{64}$$

where n represents the number of messages in a group of messages with the same identifier. j defines the bit index.

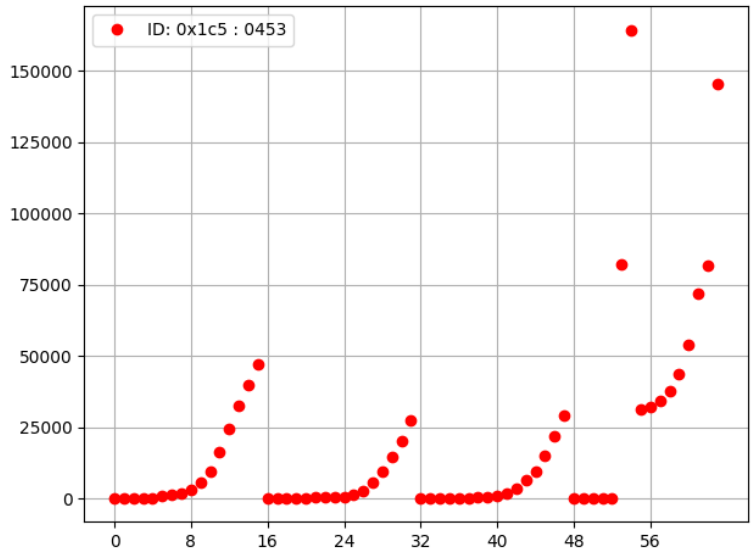

Figure 1: TAV of data bits from CAN frame group with identifier 0x1c5. The x-axis indicates the bit position inside the CAN frame. The y-axis indicates the number of bit-transitions (bit-flips).

The scatter plot of the TAV from CAN identifier 0x1c5 (figure 1) indicates some data-fields which might be contained in a CAN message of identifier 0x1c5. The exponential behavior of the transitions in the first 48 bits indicate that integer like values were transmitted in these bit areas. Another interesting behavior can be observed on the bits 53 and 54. Bit 54 has a transition in almost every CAN message. This can be seen on the relatively high number of transition. Bit 53 is has approximately half of the transitions of bit 54. These two bits together show the behavior of a multiplexer field in CAN messages. As a result of this multiplexer field, the data represented in the bit areas 55 to 63, has to be split in four individual data stream for time series analysis of this data. Another strong indicator for multiplexer fields is the predictability. The multiplexer can be treated as two-bit counter, which increments on every CAN message of identifier 0x1c5. This behavior makes it very easy to find multiplexer fields in CAN traffic. The next remarkable field behavior that can be extracted from a TAV are constant bits. Constant bits are easy to find, since the number of transition during a capture has to be zero.

To summarize the relevance of a TAV for data field recognition in CAN frames, the following statements apply:

1. Exponential behavior of bit transitions indicate integer value fields

$$y_j \approx 2y_{j+1} \tag{1}$$

2. Multiplexer fields have a width between 2 and 4 bits, usually. Let n be the number

of CAN frames in an identifier group.

$$y_j \approx n, \qquad 2y_j \approx y_{j-1} \tag{2}$$

3. Constant fields have zero transitions

$$y_j = 0 \tag{3}$$

4. Constant fields and multiplexer fields split data fields.

3.2 Derivative of a TAV

The derivative of a TAV is another useful feature for the classification of data fields in a CAN frame. A perfect example is given by the CAN identifier 0x03d. The TAV and it's derivative is shown in figure 2.

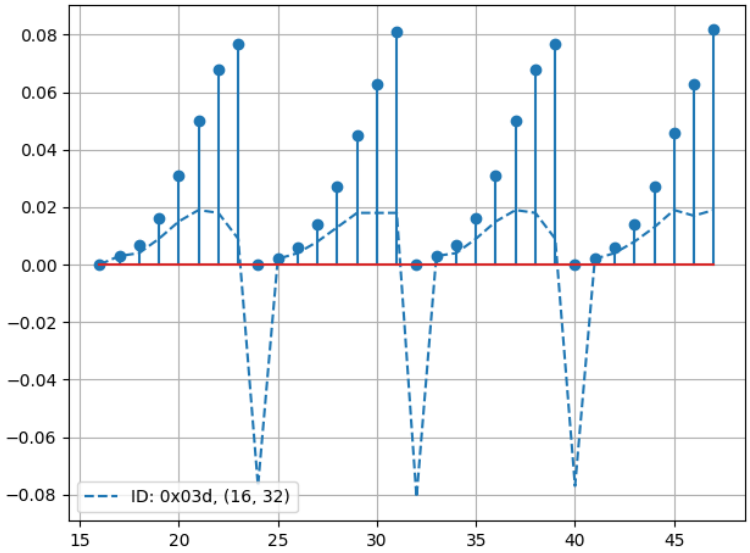

Figure 2: The derivative of a normalized TAV of data bits from the CAN frame group with identifier 0x03d. The x-axis indicates the bit position of the CAN frame. The y-axis indicates the normalized number of transition over a capture. The dashed line marks the derivative over the transitions.

An obvious feature of the derivative are negative values. These negative values are indicators for positions where a data field has to be split into sub-fields. The example in figure 2 shows the TAV of a data field from bit 16 to bit 48. The TAV strongly indicates four different 8-bit wide integer fields. The derivative (representedline from the dashed line) indicate separations after the bits 23, 31 and 39.

In real world scenarios, the TAV isn't that ideal, usually. A decision for a field separation can't be determined just by the TAV or the derivative of a TAV. A downside

of the TAV is the exclusion of the behavior of the individual bits in a time series. Two consecutive bits can behave complete individual over time. A TAV can't represent this since only the number of bit flips are counted.

3.3 Bit-Correlation-Over-Time

To fill the gap of the TAV, not be able to take the behavior in time into consideration, Enrico Pozzobon introduced a method to correlate the behavior of bits over time.

Let

$$A = \begin{pmatrix} b_{63,0} & b_{62,0} & \dots & b_{0,0} \\ b_{63,1} & b_{62,1} & \dots & b_{0,1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{63,n} & b_{62,n} & \dots & b_{0,n} \end{pmatrix} \in \mathbb{F}_2^{64 \times n} \tag{4}$$

be a Matrix, representing the time series of length n. Each point in time, row of A, can be represented as a 64 bit vector $(b_{63}, b_{62}, \dots, b_0) \in \mathbb{F}_2^{64}$. As first operation, the discrete difference over all 64 columns of A is computed.

$$B_{j,i} = (A_{j,i} \oplus A_{j,i+1})_{j=0,\dots,64 \quad i=0,\dots,n-1} \tag{5}$$

We call the result of this operation B.

$$B = \begin{pmatrix} b_{63,0} & b_{62,0} & \dots & b_{0,0} \\ b_{63,1} & b_{62,1} & \dots & b_{0,1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{63,n-1} & b_{62,n-1} & \dots & b_{0,n-1} \end{pmatrix} \in \mathbb{F}_2^{64 \times n-1} \tag{6}$$

Let m be the convolution length and

$$V = \mathbb{1} \in \mathbb{F}_2^{64 \times m} \tag{7}$$

the convolution matrix of length m. Let's define b and v as columns of B and V. The convolution between b and v can be computed through the linear discrete convolution function

$$c_i = (b_i * v_i)_{i=0,\dots,64} \tag{8}$$

The result of equation 8 applied on every column of B is stored as a matrix

$$C = \begin{pmatrix} b_{63,0} & b_{62,0} & \dots & b_{0,0} \\ b_{63,1} & b_{62,1} & \dots & b_{0,1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{63,n-1+m} & b_{62,n-1+m} & \dots & b_{0,n-1+m} \end{pmatrix} \in \mathbb{F}_2^{64 \times n-1+m} \tag{9}$$

Let the result of this discrete linear convolution function be the matrix $C_{64 \times n-1+m}$.

As last step, the Pearson correlation coefficient is applied on two consecutive columns of C in order to obtain the degree of correlation between two bits.

$$\rho_{C_i,C_{i+1}} = \frac{\mathrm{cov}(C_i, C_{i+1})}{\sigma_{C_i}\sigma_{C_{i+1}}} \qquad i = 0, \ldots, 63 \tag{10}$$

The result of this operation is a vector of length 63. Every element of this vector has a value between -1 and 1 and describes the correlation between two consecutive bits over time.

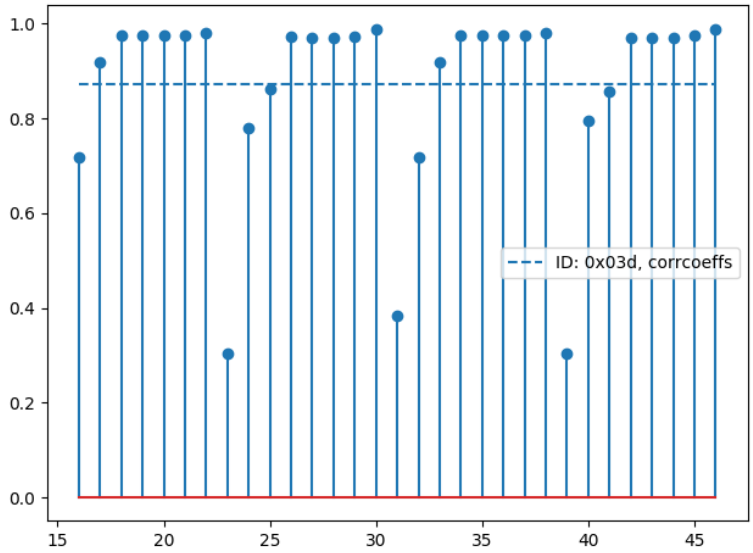

Figure 3: Bit-Correlation-Over-Time of all CAN frames of identifier 0x3d. The x-axis indicates the position between two bits. The represented field is identical to the field shown in figure 2.

3.4 TAV Derivative vs. Bit-Correlation-over-Time

Both features, the TAV Derivative and the Bit-Correlation-over-Time, have the same number of information. The combination of these two features can be used to determine interruptions in data fields and to identify positions for separations of a field into sub-fields.

A new identifier is used to discuss the combination of these two features.

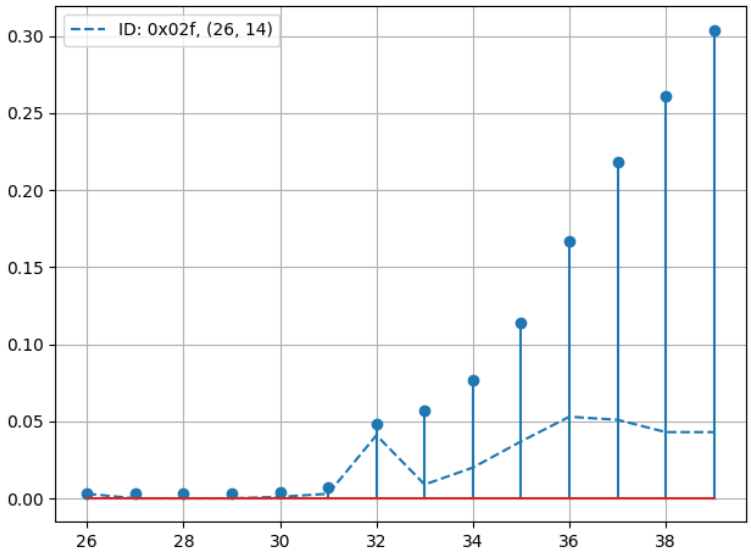

Figure 4: TAV and derivative of TAV for identifier 0x02f.

The derivative of the TAV can't be used to find a clear separation between the two sub-fields contained in data field of identifier 0x02f.

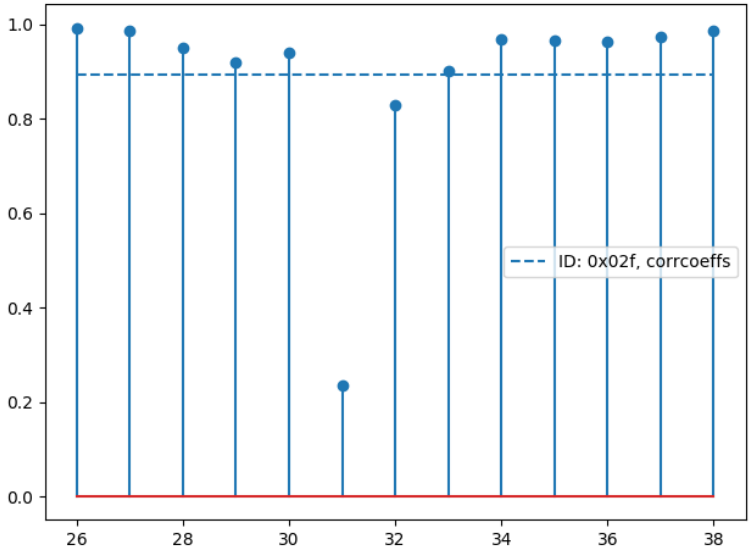

Figure 5: Bit-Correlation-Over-Time of all CAN frames of identifier 0x2f. The x-axis indicates the position between two bits.

The Bit-Correlation-Over-Time indicate a clear outlier for the position between the bits 31 and 32. It's hard to determine a sufficient threshold for the detection of separators

of value fields.

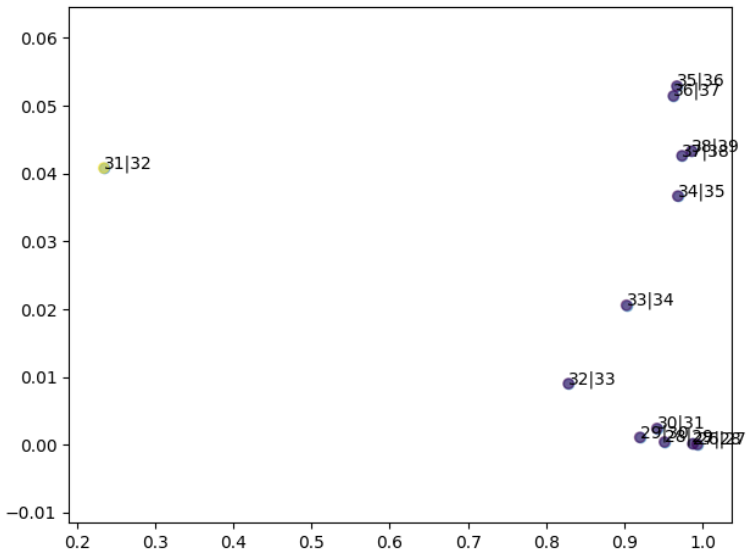

Figure 6: Derivative of TAV vs. Bit-Correlation-Over-Time of all CAN frames of identifier 0x2f.

Through the combination of these two individual features, outlier detection algorithms can be performed for the determination of separators of data fields. Figure 6 indicates a clear outlier for the position between bits 31 and 32.

3.5 Finding separator of value fields

For the identification of separators in data fields, several outlier detection methods and group finding methods are evaluated.

The scatter plot in figure 6 clearly shows two groups. Data-points which have a small value in the x-coordinate and don't form a group with other data-points are very likely separators of data fields. Which methods can be used to find outliers or groups of data-points which have a low x-coordinate.

3.5.1 Outlier Detection Methods

One approach is to treat data-points with similar x- and y-coordinates as group of data-points inside a value field. Outliers can be treated as separators, if this is applicable.

An interesting example for the evaluation of outlier detection methods is delivered by identifier 0x20.

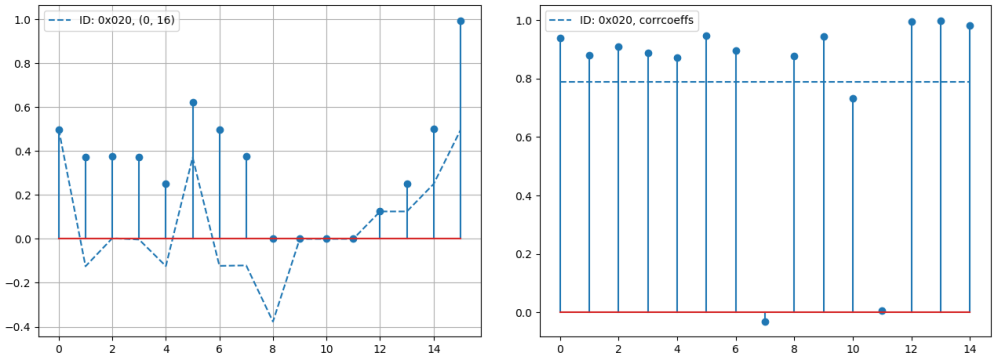

Figure 7: TAV and Bit-Correlation-Over-Time of identifier 0x20

From figure 7 a human can easily detect three fields with different kinds of data. One separation is between the bit positions 7 and 8. This separator is identified from the low value of the bit-correlation-over-time. This separator couldn't be identified from the TAV, easily. The next separation is between the bits 11 and 12. Again, the TAV wouldn't be sufficient to detect this separation.

One-Class-SVM

Unsupervised outlier detection can be performed with a One-Class-SVM.

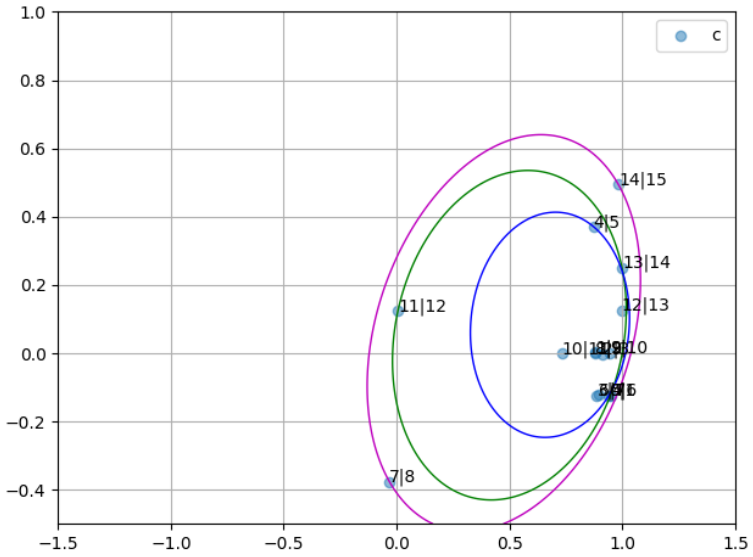

Figure 8: Scatter plot of identifier 0x20 with One-class-SVMs. The x-axis represents the Bit-Correlation-Over-Time. The y-axis represents the value of the derivative of the TAV.

Figure 8 shows the output of three different One-Class-SVMs. As described above, a human can clearly see the two outliers "7 8" and "11 12". All One-Class-SVMs use an radial basis function kernel. Different values for the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors are tested. I will call this parameter "nu". In figure 8 the magenta One-class-SVM has the "nu"-value 0.1. The green One-class-SVM has a "nu"-value of 0.2 and the blue One-class-SVM has a "nu"-value of 0.4. This example clearly shows, that a outlier detection for this two features, can't be performed by a One-class-SVM. Small "nu" values lead to over-fitting and higher values to under-fitting.

Local Outlier Factory

Another method for unsupervised outlier detection is a Local Outlier Factory. The Local Outlier Factory method is based on a k-nearest neighbors algorithm. The outlier score is represented as circle around a data point. Greater radius represents higher outlier scores. The number of neighbors for the underlying algorithm is dynamically set to half of the number of data points.

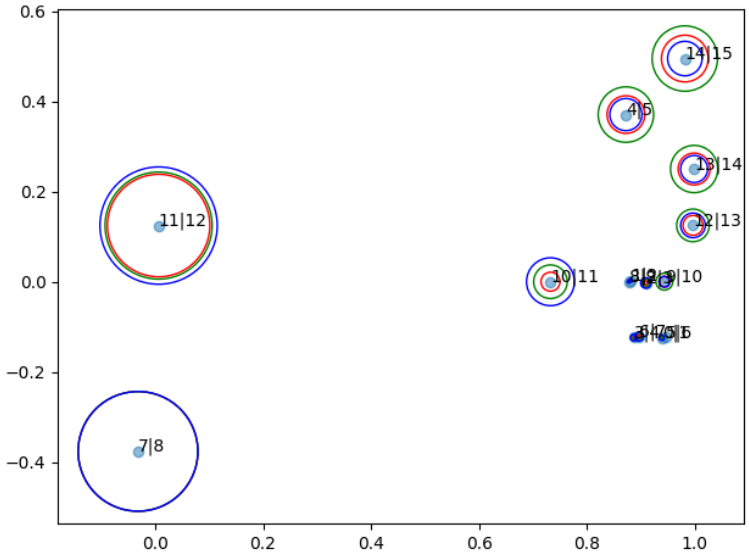

Figure 9: Scatter plot of identifier 0x20 with different Local Outlier Factories. The x-axis represents the Bit-Correlation-Over-Time. The y-axis represents the value of the derivative of the TAV.

The different colors in figure 9 represent different metrics for the k-nearest neighbor algorithm. The red Local Outlier Factory uses a squared Euclidean distance. The green Local Outlier Factory uses the Minkowski distance, and the blue Local Outlier Factory uses a weighted Minkowski distance.
Figure 9 shows great results for the detection of the searched outliers. In general, this

method has one major disadvantage. In practice, it's difficult to specify a proper threshold for the outlier score.

3.5.2 Clustering Algorithms

A different approach compared to outlier detection algorithms are clustering algorithms. If clustering algorithms can be applied on these data, two groups could be created. One group for separators, one for not separators. The group of separators can easily be identified by the values of it's members. All separators should have values close to zero, in x and y.

K-Means Algorithm

The clustering will be performed by a K-Means algorithm for two clusters.

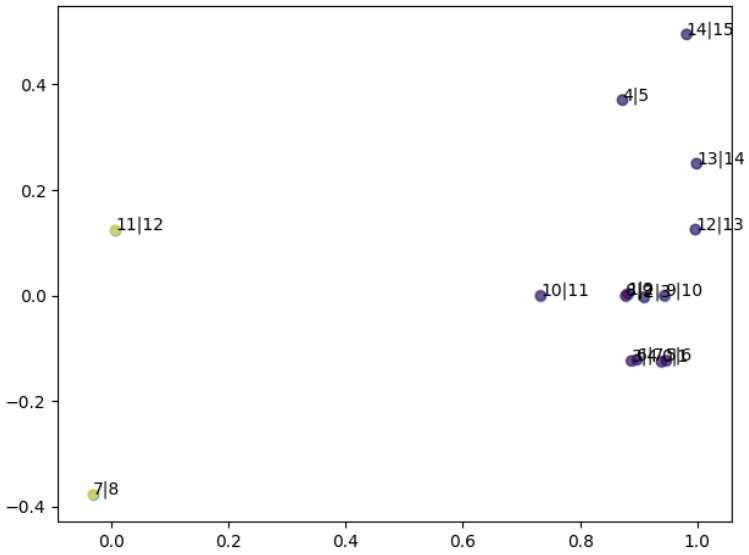

Figure 10: Scatter plot of identifier 0x20 with two clusters, computed by a K-Means algorithm. The x-axis represents the Bit-Correlation-Over-Time. The y-axis represents the value of the derivative of the TAV.

The color of the data points in figure 10 is given by K-Means algorithm. A cluster of separators is formed by the data points "7 8" and "11 12", which is the desired result.

3.5.3 Conclusion

The K-Means algorithm delivers the best result from all three evaluated algorithms. The output of the K-Means algorithm directly indicates a group of separators. The decision, which of the two output clusters of the K-Means algorithms is the separator group, and which is the non-separator group, can be made on the values of the data points. This

delivers a straight forward procedure for the separator identification.

Algorithm:

1. Check if the field has a reasonable size (for example more than four bits)
2. Compute TAV and Bit-Correlation-Over-Time of a unknown field
3. Check if the difference of minimum and maximum of the Bit-Correlation-Over-Time is reasonable big. This verifies, that separators are present
4. Apply K-Means on the data set
5. Identify the group of separators by choosing the group with lower values in the data points.

This algorithm allows to split a field into smaller fields for further evaluation of the contained data.

3.6 Linear Regression for Field extension

After the application of separation algorithm it's very likely that some fields with just on bit length are produced.

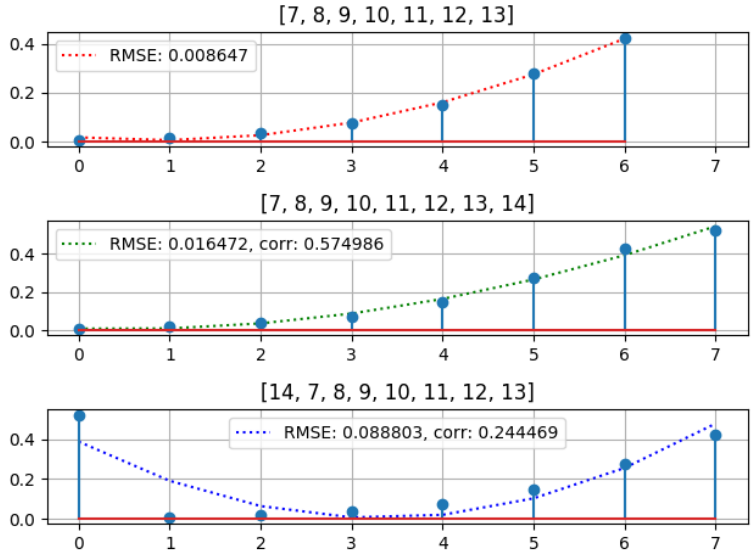

Figure 11: TAV of a field with the bit-positions 7 to 13. Bit 14 of this data frame is appended on the right and the left. RMSE of the regression and the Bit-Correlation-Over-Time (corr) between the field and the appended bit is shown.

This single bit fields might be part of a bigger field at some nearby bit-positions. In a further step, linear regression is used as a metric to detect if a single bit is part of a bigger field.

Linear regression will be applied on the TAV of a field. Since the byte-order of the communicated data is unknown a bit could be appended on the left or the right side of a field. Figure 11 shows an example. A field generated by the bits 7 to 13 and the bit 14 of the same data frame are tested through two metrics. One metric is the RMSE of a linear regression with a polynomial function of degree two. A second metric is the Bit-Correlation-Over-Time between the single-bit and the border bit of the field. Figure 11 shows an increase of the RMSE by a factor of ten compared to RMSE of the field without extension. Also the Bit-Correlation-Over-Time value is lower, if bit 14 is appended on the left side of the field. The higher Bit-Correlation-Over-Time and the lower RMSE indicate that it is possible to extend the field by appending bit 14 on the right side.

3.6.1 Limitations

This metric has some limitations. The linear regression on the TAV can only be applied, if integer values are transmitted in the field. The natural property, that least significant bits of an integer value will flip more often as most significant bits during the transmission of continuous data, is used by this metric.

If the field under investigation transmits for example values of an enumerator, this metric will not deliver any useful results.

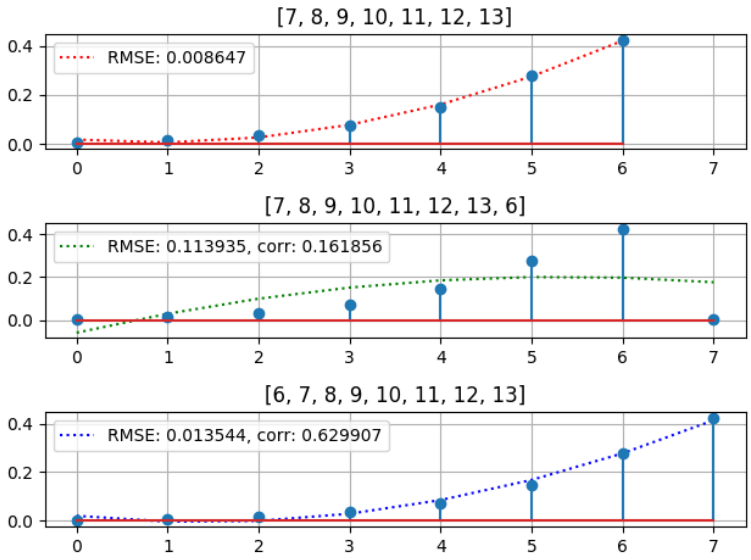

Figure 12: TAV of a field with the bit-positions 7 to 13. Bit 6 of this data frame is appended on the right and the left. RMSE of the regression and the Bit-Correlation-Over-Time (corr) between the field and the appended bit is shown.

Figure 12 shows another example. In this example bit 6 is appended on the left side of the field. The metrics clearly indicate that the bit 6 can be appended on the left. Nevertheless, this metric has it's limitations for most significant bits of signals which represent physical values. On real-world data, the most significant bits flip only a few times, which limits the information of its TAV.

3.6.2 Most Significant Bit Solution

To overcome this problem, the data over time of a field has to be analyzed. Figure 13 shows this example. The error is computed by the sum of the derivative of the normalized values of the field.

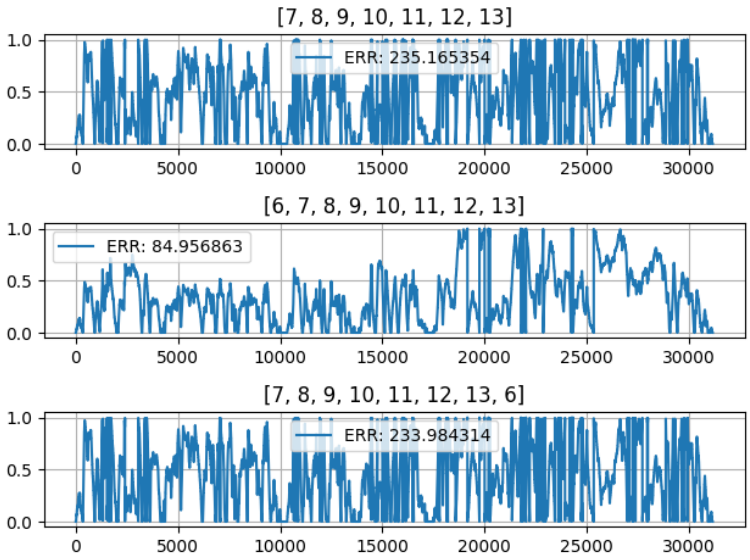

Figure 13: Values of a field over time. Bit 6 is appended on the left and on the right.

The analyzation of the error of the derivative of the data over time has a much stronger indication for the extension of most significant bits. Figure 13 shows this statement. On the second plot of this figure, the most significant bit is appended on the correct side. This reduces the error value.

4 Conclusion

The crucial part of pattern recognition on CAN communication log data is the feature extraction step. This research shows that a K-Means algorithm can be used for field separation. After separation, polynomial regression can be used to combine bits into a field. This approach of splitting and combining bits into fields delivered useful results. The combination of this algorithms can be used in an automated tool for the classification of CAN communication data. This research is used to create an open source software for CAN communication data reverse engineering.