

# Unsupervised Driving Event Discovery Based on Vehicle CAN-data

Thomas Kreutz<sup>1</sup>, Ousama Esbel<sup>2</sup>, Max Mühlhäuser<sup>1</sup> and Alejandro Sanchez Guinea<sup>1</sup>

**Abstract**—The data collected from a vehicle’s Controller Area Network (CAN) can quickly exceed human analysis or annotation capabilities when considering fleets of vehicles, which stresses the importance of unsupervised machine learning methods. This work presents a simultaneous clustering and segmentation approach for vehicle CAN-data that identifies common driving events in an unsupervised manner. The approach builds on self-supervised learning (SSL) for multivariate time series to distinguish different driving events in the learned latent space. We evaluate our approach with a dataset of real Tesla Model 3 vehicle CAN-data and a two-hour driving session that we annotated with different driving events. With our approach, we evaluate the applicability of recent time series-related contrastive and generative SSL techniques to learn representations that distinguish driving events. Compared to state-of-the-art (SOTA) generative SSL methods for driving event discovery, we find that contrastive learning approaches reach similar performance.

## I. INTRODUCTION

Deep learning is being applied successfully in the automotive industry to many use cases such as autonomous driving [1] or predictive maintenance (PdM) [2]. For instance, for PdM, multivariate time series data collected from a vehicle’s Controller Area Network (CAN) can be used to estimate the health condition of a car component [2]. Furthermore, as CAN-data reflects the overall state of the vehicle, based on this data, it is possible to infer information about the current driving event, situation, or behavior (e.g., [3], [4]).

A car can generate multiple gigabytes of vehicle CAN-data in a single driving session [5], which may grow quickly to multiple tera- or petabytes when considering fleets of cars. For this reason, it is a time-consuming and expensive task for domain experts to manually annotate or analyze these enormous amounts of data [4]. Consequently, problems like the sampling bias may limit the performance of, for instance, PdM models trained on non-reviewed data. A possible solution is using unsupervised machine learning to discover information in extensive unlabelled collections of data. Recent work has used self-supervised learning (SSL) to transform unlabelled vehicle CAN-data into a new representation that allows distinguishing different driving events or behavior (e.g., [3], [6]) without any prior knowledge.

This paper focuses on the unsupervised discovery of driving events by proposing a method that simultaneously clusters and segments vehicle CAN-data. Consider the driving events A, B, and C composing a driving scenario in Fig. 1.

<sup>1</sup>The authors are with the Telecooperation Lab at the Technical University Darmstadt, Germany {kreutz, max, sanchez}@tk.tu-darmstadt.de

<sup>2</sup>The authors are with COMPREDICT GmbH, Darmstadt, Germany {esbel}@compredict.de

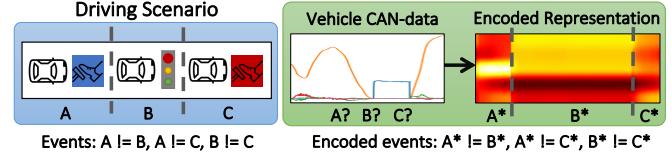


Fig. 1. A neural network encodes vehicle CAN-data into a representation, where the same driving events become similar at all time steps. Such a representation allows to cluster each time step in isolation, which simultaneously clusters and segments vehicle CAN-data into driving events.

State-of-the-art (SOTA) SSL methods for vehicle CAN-data (e.g. Drive2Vec [3]) learn representations of vehicle CAN-data, where all time steps that are part of, e.g., event type A share a similar representation  $A^*$ , and all time steps of other events are different ( $B^*$ ,  $C^*$ ). Due to this property in the encoded representation, we can cluster all individual time steps in isolation and obtain a state sequence that is 1) constant for the duration of events and 2) only differs at the event changepoints, which effectively segments and clusters the data the same time.

Previous work on unsupervised driving event discovery can be classified into motif discovery (e.g., [7]), time series segmentation (e.g., [8], [4]), subsequence clustering (e.g., [9], [10]), simultaneous segmentation and clustering (e.g., [11]) and generative SSL methods for representation learning (e.g., [6], [3]). Aside from the latter SSL methods and the approaches in ([9], [11]), the majority of previous methods first uses a segmentation or motif discovery algorithm to discover potential event segments, extracts segment level features with, e.g., a neural network, and then clusters the resulting segments in a mandatory final step.

Similar to and inspired by Toeplitz Inverse Covariance-based Clustering (TICC) [11] and Variational Animal Motion Embedding (VAME) [12], our approach simultaneously clusters and segments vehicle CAN-data. Because our approach produces a sequence of event states that are split into segments at their changepoints, it is not limited to a maximum length of events as in [9] and is independent of segments obtained by a segmentation or motif discovery algorithm.

Furthermore, we evaluate recent contrastive SSL methods for representation learning of vehicle CAN-data. To the best of our knowledge, we are the first to explore contrastive SSL methods for this kind of data. Recent advances in deep learning-based time series changepoint detection (CPD) [13] and contrastive SSL for multivariate time series (e.g., [14], [15]) motivate to research the applicability of contrastive SSL for vehicle CAN-data. Contrastive SSL has shown extraordinary results in distinguishing different states when

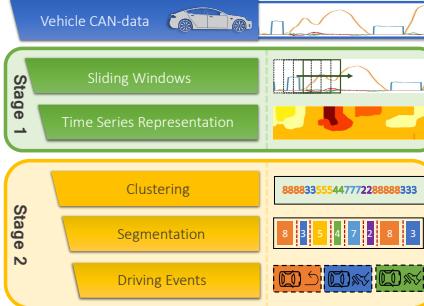


Fig. 2. High level overview of the proposed two-stage approach

applied to human activity recognition (HAR) data [13], electroencephalography (EEG) data [15], or household consumption data [14], among others. The main contributions of this work are as follows:

- An approach to discover driving events in an unsupervised manner, which simultaneously clusters and segments vehicle CAN-data.
- An empirical evaluation of contrastive SSL methods for vehicle CAN-data, which is (to the best of our knowledge) the first step in the direction of applying these kinds of methods to vehicle CAN-data.

## II. NOTATIONS AND DEFINITIONS

We define a multivariate time series  $MTS$  as a tuple  $MTS := (X_1, X_2, \dots, X_N) \in \mathbb{R}^{d \times N}$  where each element  $X_t \in MTS$  represents a  $d$ -dimensional measurement such that  $X_t = (x_1, \dots, x_d)$  for all  $0 < t \leq N$ .

We define a subsequence  $S$  of  $MTS$  as a collection of points starting at a time point  $k$  and ending at index  $l$ . Formally, we define this subsequence as  $S_{k,l} = (X_k, \dots, X_l) \subseteq MTS$  where  $\subseteq$  denotes that  $S$  is a subsequence of  $MTS$ .

A sliding window is a fixed size window of length  $w$  that moves sequentially over  $MTS$ . For an  $MTS$  of length  $N$  and for each time point  $X_t \in MTS$  with  $w \leq t \leq N$ ,  $w \ll N$ , we can define a sliding window as a subsequence of this  $MTS$  with  $S_{t-(w-1),t} \subseteq MTS^1$ .

## III. APPROACH

A high-level overview of our two-stage approach is depicted in Fig. 2. In Stage 1, we transform vehicle CAN-data into a new representation by using the sliding window technique and encoding each sliding window with a neural network. The encodings are concatenated to compose a new time series which is enriched at each time step with information about the current driving event state. Afterward, in Stage 2, we cluster the sliding window representations and obtain a one-dimensional cluster sequence, which can be considered a discrete driving event state sequence. The state sequence is segmented in variable-length driving events

<sup>1</sup>Any timepoint  $l = t$  is the end of a sliding window, and  $k = t - (w - 1)$  is the start. The first possible sliding window always starts at  $k = 1$  and ends at  $t = w$  and the last sliding window always ends with  $l = N$ . This reduces the time series by  $w - 1$  time steps, and  $w$  is the new start point.

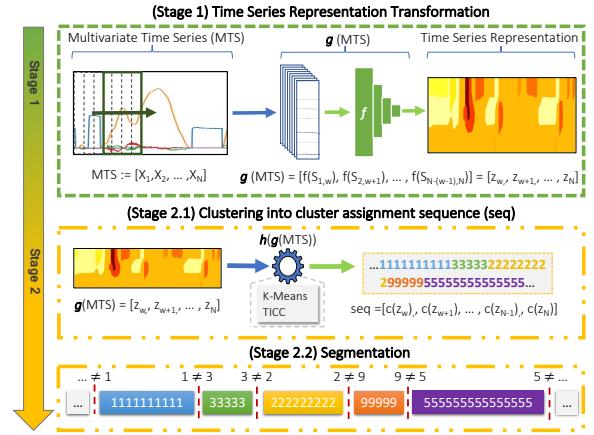


Fig. 3. Detailed overview of both stages of the approach. Stage 1 concerns with the time series representation, Stage 2 concerns with the clustering and segmentation

by splitting the sequence at the cluster changepoints. The corresponding constant cluster of a segment characterizes the respective driving event.

### A. Stage 1 - Time Series Representation

We summarize our approach in more detail in Fig. 3. The *time series representation* in Stage 1 is obtained by first sliding a fixed-size window of length  $w$  over a  $d$ -dimensional multivariate time series  $MTS \in \mathbb{R}^{d \times N}$  of length  $N$ . Afterward, each window is encoded to a new representation. The resulting representations are concatenated to a new multivariate time series that has a different dimension but preserves the temporal order.

We consider a representation to be a mapping from a  $d \times w$ -dimensional sliding window input to a new  $e$ -dimensional representation. This mapping is defined by an encoder function  $f : \mathbb{R}^{d \times w} \rightarrow \mathbb{R}^e$ . The function  $f$  can be interpreted as the encoder part of an Autoencoder (AE). For  $f$ , we use the concept of a *timestamp-level representation*, where each point in time  $X_t \in MTS$  with  $w \leq t \leq N$  is mapped to a new representation  $z_t$ . We refer to the representation  $z_t = f(S_{t-(w-1),t})$  as a *timestamp-level representation*, which is the encoding of a sliding window  $S_{t-(w-1),t}$ .

We define a *time series representation* as an  $e$ -dimensional representation of a  $d$ -dimensional  $MTS$ , where each sliding window was encoded with  $f$ . Formally, the *time series representation* is a function  $g : \mathbb{R}^{d \times N} \rightarrow \mathbb{R}^{e \times N-(w-1)}$  that maps (encodes) the elements of  $MTS$  to an  $e$ -dimensional representation and preserves their temporal order. By applying  $f$  to all sliding windows over  $MTS$ , we obtain the *time series representation* function  $g$ , defined as:

$$g(MTS) = [f(S_{1,w}), f(S_{2,w+1}), \dots, f(S_{N-(w-1),N})] \quad (1)$$

$$= [z_w, z_{w+1}, \dots, z_N] \quad (2)$$

### B. Stage 2 - Simultaneous Segmentation and Clustering

A detailed overview of Stage 2 is depicted in Fig. 3. In Stage 2.1, each *timestamp-level representation*  $z \in [z_w, z_{w+1}, \dots, z_N] = g(MTS)$  is clustered, with the

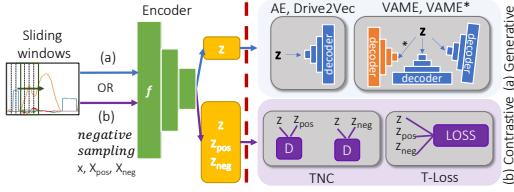


Fig. 4. Overview of the neural networks used in this work. a) encoder-decoder models, b) contrastive learning models

number of clusters fixed to  $k$ . The clustering process can be formally defined by a partition  $h := \{H_1, \dots, H_k\}$  that partitions a *time series representation*  $z$  to its cluster by a function  $c : \mathbb{R}^e \rightarrow \mathbb{N}$ . For any  $z \in \mathbb{R}^e$ ,  $c(z)$  returns the index  $j$  of the cluster where  $z \in H_j$ . We apply  $c$  to each  $z \in [z_w, \dots, z_N]$  and obtain a sequence of cluster assignments  $seq = [c(z_w), \dots, c(z_N)]$ .

$$\forall z \in \mathbb{R}^e. c(z) = j \implies z \in H_j \quad (3)$$

In Stage 2.2., a simple segmentation step splits the cluster assignment sequence  $seq$  at the changepoints into segments. Finally, each segment is assigned to its constant cluster assignment, which directly clusters the obtained segments.

*Driving Event Discovery:* The discovery of driving events from the segmentation is achieved by grouping all segments under their corresponding constant cluster assignment. The segments of a cluster can be interpreted by, e.g., a domain expert, or they can be compared to labeled data to find high overlaps between a cluster and ground truth annotations. Finally, each segment in the time series can be annotated with the driving event interpretation of its cluster. The segmentation can be visualized, for example, on a GPS trajectory by coloring each time step with its cluster assignment.

### C. Application to Multiple Multivariate Time Series

Our approach can easily scale to multiple *MTS* when considering Stage 1 with a collection of multivariate time series ( $C_{MTS}$ ) instead of a single *MTS*. In this setting, Stage 1 is carried out for each  $MTS \in C_{MTS}$  individually and we obtain a collection of time series representations  $C_z := \{g(mts) \mid mts \in C_{MTS}\}$ . In Stage 2.1, the partition  $h$  is computed on all datapoints in  $C_z$ . Finally, we can obtain a collection of cluster assignment sequences  $C_{seq} := \{[c(z_a), \dots, c(z_b)] \mid [z_a, \dots, z_b] \in C_z\}$ .

### D. Neural Network Encoder Training

Following the SSL taxonomy in [16], the encoder  $f$  can be trained in a generative or a contrastive manner. A generative (Encoder-Decoder) model is usually trained with a decoder  $f_{dec}$  that has the goal to reconstruct the  $d$ -dimensional input from the  $e$ -dimensional representation of  $f$  with minimal error. Contrastive learning directly trains the encoder with a contrastive loss that pushes similar data points

to the same representation and non-similar data points to a different representation.

As summarized in Fig. 4, as generative methods, we use a normal AE as a baseline that was used in [17], and a Variational AE (VAE) with an additional future decoder and  $k$ -means objective, i.e., VAME [12]. Furthermore, we compare these methods to a SOTA Encoder-Decoder model for vehicle CAN-data called Drive2Vec [3]. Drive2Vec learns representations that are predictive about the future, helping to better differentiate the current driving situation. Similarly, VAME is trained with a present and a future decoder to learn representations that are predictive about the future. We extend VAME to VAME\* with a past decoder to learn predictive representation about the past and the future.

As contrastive learning methods, we consider Temporal Neighborhood Coding (TNC) [15] and a triplet loss approach (T-Loss) [14]. TNC assumes a time series to be stationary for the full duration of an event. Given a reference window, they propose estimating its stationary temporal neighborhood with a statistical test. Afterward, random windows from this neighborhood are enforced to be encoded to a similar representation, and windows outside the neighborhood to a different one by using a discriminator on the embeddings. T-Loss accepts variable-length time series input and is trained with a triplet loss. Random subsequences of a reference window are enforced to be encoded similarly to it, but different to other random windows. Similar to TNC, a larger reference window can be seen as an event's temporal neighborhood.

More technical details about the selected SSL methods are out of scope in this manuscript and we must point to the original works for further reading. The SSL methods we consider can encode corresponding time steps similar to each other. Towards this end, TNC has been used for, e.g., HAR data [15], T-Loss for Household Consumption data [14], and an AE in [17] for CAN-data. Using future context for training has been shown to effectively encode events to similar feature representations by Drive2Vec in [3] for CAN-data, and with VAME for mice behavior data in [12]. We focus on empirically evaluating how these methods compare as a basis for our approach when learning timestamp-level representations of CAN-data.

### E. Clustering Methods

As shown in Fig. 3, we use the  $k$ -means and TICC clustering algorithms to cluster the encoded time steps of vehicle CAN-data. We chose the  $k$ -means algorithm as a method that does not include any temporal dependency between time steps in order to leverage the learned representations that distinguish different driving events. On the other hand, we chose TICC because it incorporates a temporal dependency between time steps by using a dependency network.

## IV. EXPERIMENTAL EVALUATION

### A. Experimental Setup

We use the neural networks and TICC implementations available in their respective GitHub repositories (TNC<sup>2</sup>, T-

<sup>2</sup>[https://github.com/sanatonek/TNC\\_representation\\_learning](https://github.com/sanatonek/TNC_representation_learning)

TABLE I  
RESULTS OF THE BEST PERFORMING PARAMETER CONFIGURATIONS FOR EACH MODEL, G=GENERATIVE, C=CONTRASTIVE

	VAME* (G)		VAME (G)		Drive2Vec (G)		T-Loss (C)		TNC (C)		AE (G)	
	<i>k</i> -means	TICC										
$F_1$ (↑)	<b>0.469</b>	0.399	0.464	0.374	0.430	0.378	0.392	0.377	0.338	0.294	0.310	0.286
$e$	20	20	20	20	15	10	5	15	5	20	20	10
$w$	10	5	15	10	10	5	10	15	16	10	5	10

Loss<sup>3</sup>, VAME<sup>4</sup>, TICC<sup>5</sup>). As suggested in [18], we have modified the TNC model to use an encoder with dilated causal convolutions. For Drive2Vec, we use our implementation that, instead of gated recurrent units (GRU), uses a temporal convolutional network (TCN). We implemented a version of the AE from [17] in the PyTorch<sup>6</sup> deep learning framework.

### B. Use Case and Training Dataset

We consider a use case from the automotive domain, where the goal is to discover frequently occurring driving events in vehicle CAN-data in an unsupervised manner. We have collected a dataset that consists of 60 multivariate time series recordings of a Tesla Model 3 vehicle. The data has been recorded in real driving sessions by using a datalogger. The datalogger is mounted into the car to directly access the CAN-bus over an external interface. It records the values of 113 vehicle signals upsampled at a frequency of 200hz. However, from these 113 signals, we select only a subset of nine signals that we consider to be relevant for driving events: ‘Brake pressure front left/right’ [bar], ‘Electric motor torque’ [Nm], ‘Accelerator pedal position’ [%], ‘Steering wheel angle’ [°], ‘Velocity’ [km/h], ‘Longitudinal acceleration’ [m/s<sup>2</sup>], ‘Lateral acceleration’ [m/s<sup>2</sup>], and ‘Yaw rate’ [°/s]. To preprocess the data, we follow related work (e.g., [3], [6], [11]) and resample the data from 200Hz to 10Hz and apply Z-normalization separately on each channel.

### C. Ground Truth - Manual Annotation of Testing Data

We have manually annotated a driving trip to compare our approach’s results against a ground truth. The trip was around two hours long (following the evaluation in [11]), and designed to include different traffic scenarios that we believe to occur on urban, highway, or secondary roads. For the annotation we considered 13 different labels: ‘Standstill’, ‘Acceleration’, ‘Deceleration’, ‘Turn Left’, ‘Turn Right’, ‘Braking’, ‘Keep Velocity (KV)’, ‘Fast KV’, ‘Autonom KV’, ‘S Turn’, ‘Reverse’, ‘Lane Change’, and ‘Roundabout’.

### D. Implementation Details

Regarding the neural networks, we follow [14] and do not perform any hyperparameter optimization on the network parameters. We evaluate our approach’s general hyperparameters: The window size  $w$  and the embedding dimension  $e$ . In particular, we evaluate  $w \in [5, 10, 15, 20]$  and

<sup>3</sup><https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries>

<sup>4</sup><https://github.com/LINCellularNeuroscience/VAME>

<sup>5</sup><https://github.com/davidhallac/TICC>

<sup>6</sup><https://pytorch.org/>

$e \in [3, 5, 10, 15, 20]$ . Each model is trained for 200 epochs (800 steps for T-Loss) with a learning rate of  $1e-4$ . Sliding windows with a step size of 3 are used for training, and a step size of 1 is used at the clustering and testing step. T-Loss is trained with a reference window that is  $3 \times w$ .

Concerning the clustering algorithms, the number of clusters is fixed to  $k = 13$ , which matches the number of ground truth labels. The *k*-means algorithm is run 15 times with random restarts. For the TICC<sup>7</sup> algorithm we use the parameters *window\_size* = 10,  $\lambda = 5 - e3$ , *beta* = 400, *max\_it* = 3, *thresholds* =  $2e-5$ .

### E. Best Performing Approach Configurations

We summarize the results for the best performing models for both clustering algorithms in Table I, which also shows the corresponding parameters  $e, w$  in the bottom rows. The row  $F_1$  summarizes the macro  $F_1$  score over all classes. On average, VAME\* and VAME outperform the other models. Drive2Vec follows closely, and AE performs the worst. T-Loss and TNC fall short behind the generative models. We will see in the following sections that this performance difference is due to the ‘Turn Right’ events. For all other driving events, the performance is similar. In addition, from the results of the bottom  $e$  and  $w$  rows in Table I, we can see that a higher embedding dimension of 10-20 and a window size between 5-15 give the best results.

### F. Latent Space

The latent spaces of the best-performing models are shown in Fig. 5. In the top row, we depict the ground truth and the predictions on the bottom. We notice qualitatively that the majority of the ground truth classes are separated in the latent space, and the unsupervised clustering prediction matches them. The models can learn representations for data points of the same driving event that can be separated by the *k*-means algorithm at a timestamp level. For T-Loss and TNC, we can observe that the ‘Turn Right’ event has no matching cluster in the prediction (compare dashed circles), which is an indication why their macro  $F_1$  score in Table I is lower than for the generative models.

### G. Segmentation

In Fig. 6, we depict GPS trajectories of driving scenarios colored with a ground truth annotation (top row) and the segmentation (bottom row) of our best model w.r.t. Table I. In Scenario 1 (left), the vehicle exits a parking lot. In Scenario 2

<sup>7</sup>The original work [11] suggested ten or more iterations. The results with three iterations are promising, and we consider running more iterations in the future. More iterations can significantly improve the result.

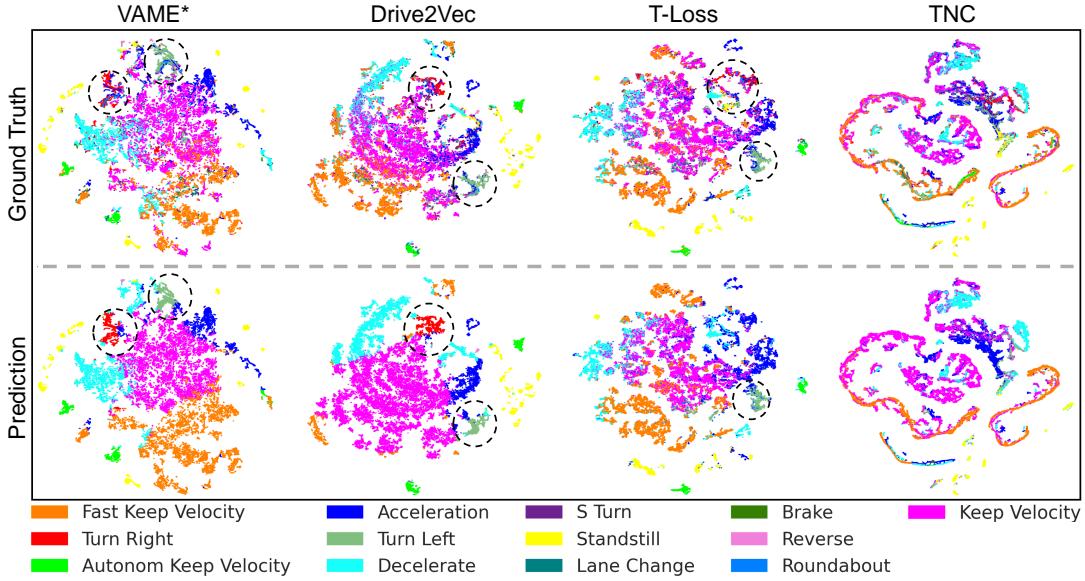


Fig. 5. Latent space visualization, where 20 dimensional representations of VAME\* have been reduced to two dimensions with TSNE. Ground truth (top), and clustering (bottom), mapped to the ground truth class with the highest overlap (best viewed in color).

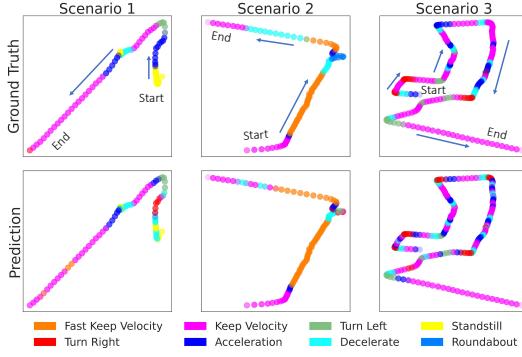


Fig. 6. Segmentation results under different driving scenarios. Ground truth (top), and clustering (bottom), predicted clusters are mapped to the ground truth class with the highest overlap

(middle), it drives a secondary road with a roundabout and enters a village. In Scenario 3 (right), it experiences a common stop-and-go scenario in the neighborhood of a city.

We observe a high similarity in our predictions compared to the ground truth. In Scenario 1, we notice that the first right turn of the vehicle was annotated as an acceleration, but our model predicts a right turn. Because both events are correct, our model can support a human annotator. In Scenario 2, our model cannot capture the roundabout in one cluster, but it predicts corresponding turns. For Scenario 3, the approach is accurate in predicting the driving event states of the car. We notice that our prediction is sometimes more accurate than our annotation, which shows a) that an annotation process can be ambiguous and b) that our approach can support a human annotator during that process.

#### H. Classification

We train a linear SVM with 10-fold cross-validation to evaluate the linear separability of the representations and the

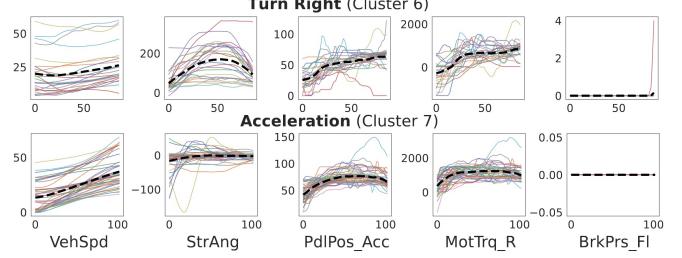


Fig. 7. Exemplary clusters and their segments with real-world interpretation

results are shown in Table II. The macro  $F_1$  results show that both generative and contrastive models achieve similar performance, with the T-Loss model having a slightly better score on average. T-Loss achieves comparable results for most events when compared to the SOTA method Drive2Vec. TNC is outperformed by T-Loss due to its neighborhood estimation approach that is not well designed for CAN-data.

The events ‘Standstill’, ‘Acceleration’, ‘Deceleration’, ‘KV’, and ‘Fast KV’ are distinguished the best in the latent space. This result also reflects the clustering predictions of the latent space in Fig. 5. For ‘Autonom KV’, ‘Turn Left’, ‘Turn Right’ and ‘Reverse’, we see a drop in the scores, which shows that the models perform worse for these events. The ‘Brake’ event is not found because its encoded similarly to ‘Deceleration’ due to an electric vehicle’s recuperation mechanism. Furthermore, the events that are composed of other events like ‘Roundabout’ can not be discovered with our approach because the window size is too small and our approach finds the events that compose it.

#### I. Simultaneous Clustering and Segmentation

We verify the segmentation ability of our approach, in Fig. 7, with all segments of two arbitrary clusters. They are visualized with five different vehicle CAN-data signals to

TABLE II

SUPERVISED MACRO  $F_1$  CLASSIFICATION RESULTS FOR EACH INDIVIDUAL CLASS ON TEST TRIP, G=GENERATIVE, C=CONTRASTIVE

	Standstill	Acceleration	Deceleration	KV	Fast KV	Autonom KV	Turn Left	Turn Right	Reverse	S Turn	Lane Change	Roundabout	Brake	Macro $F_1$
AE (G)	<b>0.839</b>	0.359	0.630	0.463	0.537	0.188	0.341	0.222	0.322	-	<b>0.014</b>	-	<b>0.025</b>	0.337
VAME* (G)	0.749	0.552	0.672	0.719	0.705	0.218	0.345	<b>0.223</b>	<b>0.662</b>	-	-	<b>0.047</b>	0.010	0.416
VAME (G)	0.764	<b>0.593</b>	<b>0.683</b>	0.715	0.679	<b>0.324</b>	<b>0.387</b>	0.161	0.446	-	-	0.026	-	0.417
Drive2Vec (G)	0.710	0.587	0.669	<b>0.731</b>	0.671	0.280	0.348	0.154	0.458	-	-	0.045	-	0.412
T-Loss (C)	0.742	0.471	0.652	0.676	<b>0.717</b>	0.204	0.320	0.035	-	-	-	-	-	<b>0.438</b>
TNC (C)	0.801	0.535	0.666	0.679	0.447	-	0.251	-	0.017	-	-	-	-	0.386

interpret the events. From left to right, these channels are ‘Velocity’, ‘Steering wheel angle’, ‘Accelerator pedal position’, ‘Motor torque’, and ‘Brake pressure’. We only show events longer than three seconds, because driving events usually last four to six seconds [19]. Further, each segment is stretched to the length of the longest segment in its cluster. The dashed black line is the average of all segments.

Each segment in the ‘Acceleration’ cluster in Fig. 7 follows a similar pattern: The vehicle’s velocity increases, the brake is not pressed, the accelerator pedal is pressed, and the motor torque is at a high value. Similarly, in the ‘Turn Right’ cluster, the steering angle increases up to 200 degrees, which signals a clear right turn. The similar segments in both clusters verify that our approach can simultaneously cluster and segment vehicle CAN-data in an unsupervised manner.

## V. CONCLUSION & FUTURE WORK

We present an unsupervised approach that discovers driving events in vehicle CAN-data. Using self-supervised learning (SSL), we encode all time steps of driving events from the same category to a similar representation. This property allows clustering each time step in isolation, which simultaneously clusters and segments vehicle CAN-data into driving events. Our experiments show the effectiveness of our approach in unlabelled CAN-data from a Tesla Model 3 vehicle. Further, we find that representations from contrastive SSL methods perform comparably to state-of-the-art generative SSL methods for CAN-data.

Possible extensions of our approach include discovering different driver’s behavior patterns for events or sequences of events with, for instance, a language model. In addition, we can improve our approach with better representation learning methods. For instance, we consider developing a better neighborhood estimation method for TNC that better captures the non-stationary nature of events in CAN-data. Furthermore, we believe that our approach is generally applicable to all kinds of multivariate time series data composed of sequences of events. We intend to evaluate the approach for vehicles with a combustion engine or even in other domains (e.g., HAR, EEG) in the future.

## REFERENCES

- [1] A. Luckow, K. Kennedy, M. Ziolkowski, E. Djerekarov, M. Cook, E. Duffy, M. Schleiss, B. Vorster, E. Weill, A. Kulshrestha, *et al.*, “Artificial Intelligence and Deep Learning Applications for Automotive Manufacturing,” in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 3144–3152.
- [2] S. Foulard, S. Rinderknecht, and R. Fietzek, “Fahrzeugkomponenten in Echtzeit überwachen,” *atp edition*, vol. 59, no. 06, pp. 54–62, 2017.
- [3] D. Hallac, S. Bhooshan, M. Chen, K. Abida, J. Leskovec, *et al.*, “Drive2vec: Multiscale State-Space Embedding of Vehicular Ssensordata,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3233–3238.
- [4] A. Bender, G. Agamennoni, J. R. Ward, S. Worrall, and E. M. Nebot, “An Unsupervised Approach for Inferring Driver Behavior From Naturalistic Driving Data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3325–3336, 2015.
- [5] K. B. Singh and M. A. Arat, “Deep Learning in the Automotive Industry: Recent Advances and Application Examples,” *arXiv preprint arXiv:1906.08834*, 2019.
- [6] H. Liu, T. Taniguchi, Y. Tanaka, K. Takenaka, and T. Bando, “Visualization of Driving Behavior Based on Hidden Feature Extraction by Using Deep Learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2477–2489, 2017.
- [7] M. I. Silva and R. Henriques, “Tripmd: Driving Patterns Investigation via Motif Analysis,” *Expert Systems with Applications*, vol. 184, p. 115527, 2021.
- [8] K. Takenaka, T. Bando, S. Nagasaka, T. Taniguchi, and K. Hitomi, “Contextual Scene Segmentation of Driving Behavior Based on Double Articulation Analyzer,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- [9] O. Shouno, “Deep Unsupervised Learning of a Topological Map of Vehicle Maneuvers for Characterizing Driving Styles,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2917–2922.
- [10] M. Siami, M. Naderpour, and J. Lu, “A Mobile Telematics Pattern Recognition Framework for Driving Behavior Extraction,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1459–1472, 2020.
- [11] D. Hallac, S. Vare, S. Boyd, and J. Leskovec, “Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 215–223.
- [12] K. Luxem, F. Fuhrmann, J. Kürsch, S. Remy, and P. Bauer, “Identifying Behavioral Structure From Deep Variational Embeddings of Animal Motion,” *BioRxiv*, 2020.
- [13] S. Deldari, D. V. Smith, H. Xue, and F. D. Salim, “Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding,” in *Proceedings of the Web Conference 2021*, 2021, pp. 3124–3135.
- [14] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, “Unsupervised Scalable Representation Learning for Multivariate Time Series,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [15] S. Tonekaboni, D. Eytan, and A. Goldenberg, “Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding,” *International Conference on Learning Representations*, 2021.
- [16] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, “Self-Supervised Learning: Generative or Contrastive,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- [17] K. Sama, Y. Morales, H. Liu, N. Akai, A. Carballo, E. Takeuchi, and K. Takeda, “Extracting Human-Like Driving Behaviors from Expert Driver Data Using Deep Learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9315–9329, 2020.
- [18] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, and B. Xu, “Learning Timestamp-Level Representations for Time Series with Hierarchical Contrastive Loss,” *arXiv preprint arXiv:2106.10466*, 2021.
- [19] X. Zhang, X. Zhao, and J. Rong, “A Study of Individual Characteristics of Driving Behavior Based on Hidden Markov Model,” *Sensors & Transducers*, vol. 167, no. 3, p. 194, 2014.