# An Interpretable Lane Change Detector Algorithm based on Deep Autoencoder Anomaly Detection

Oliver De Candido, Maximilian Binder, and Wolfgang Utschick

The 32nd IEEE Intelligent Vehicles Symposium
July 11–15, 2021
Nagoya University, Nagoya, Japan
Virtual Conference

TUM Department of Electrical and Computer Engineering
Professur für Methoden der Signalverarbeitung
Univ.-Prof. Dr.-Ing. Wolfgang Utschick

# An Interpretable Lane Change Detector Algorithm based on Deep Autoencoder Anomaly Detection

Oliver De Candido, Maximilian Binder, and Wolfgang Utschick

*Abstract*— **In this paper, we address the challenge of employing Machine Learning (ML) algorithms in safety critical driving functions. Despite ML algorithms demonstrating good performance in various driving tasks, e.g., detecting when other vehicles are going to change lanes, the challenge of validating these methods has been neglected. To this end, we introduce an interpretable Lane Change Detector (LCD) algorithm which takes advantage of the performance of modern ML-based anomaly detection methods. We independently train three Deep Autoencoders (DAEs) on different driving maneuvers: lane keeping, right lane changes, and left lane changes. The lane changes are subsequently detected by observing the reconstruction errors at the output of each DAE. Since the detection is purely based on the reconstruction errors of independently trained DAEs, we show that the classification outputs are completely interpretable. We compare the introduced algorithm with black-box Recurrent Neural Network (RNN)-based classifiers, and train all methods on realistic highway driving data. We discuss both the costs and the benefits of an interpretable classification, and demonstrate the inherent interpretability of the algorithm.**

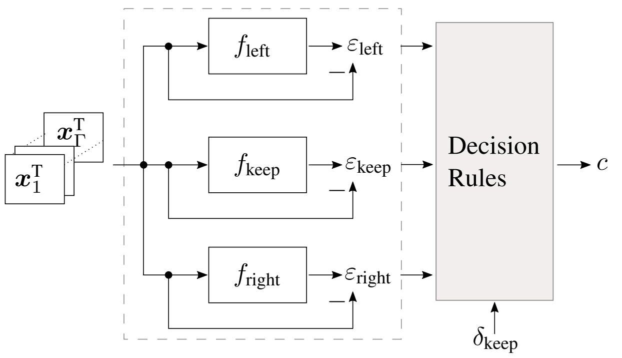

$\boldsymbol{x}_\Gamma^\mathrm{T}$
$\boldsymbol{x}_1^\mathrm{T}$
$f_\text{left}$
$f_\text{keep}$
$f_\text{right}$
$\varepsilon_\text{left}$
$\varepsilon_\text{keep}$
$\varepsilon_\text{right}$
$-$
Decision Rules
$c$
$\delta_\text{keep}$

Fig. 1: An overview of the LCD algorithm demonstrating a decoupling of the ML-based reconstruction (dashed box) and the explicit decision rules. Each function $f$ is an independently trained ML method used to reconstruct the time-series signals $\boldsymbol{x}_1^\mathrm{T}, \dots, \boldsymbol{x}_\Gamma^\mathrm{T}$. An interpretable decision rule-set outputs the decision $c$ based on the reconstruction errors.

## I. Introduction and Motivation

When we consider the ever increasing complexity of driving scenarios which Autonomous Vehicles (AVs) must handle, it becomes clear that Machine Learning (ML) algorithms will be part of the solution. However, if we think about the black-box nature of these ML algorithms, they lack the interpretability required to validate the driving functions which they are incorporated into. For example, if the classification output of an ML-based component leads to an accident, it is important to understand why that output arose. To tackle this lack of interpretability, the research community has focused on Explainable Artificial Intelligence (XAI) [1], [2], [3]. Recently, AV researchers have proposed XAI methods in various driving functions, e.g., in [4], the authors introduce a steering angle control algorithm based on driving video data. They provide visual interpretability using attention heat maps, which highlight regions in the 2D-image that might influence the algorithm's decision. In [5], the authors propose the use of interpretable representations in an end-to-end ML-based motion planner. These are used to visualize the motion forecasting and to quantify the object detection performance.

In this paper, we consider the example driving function to detect when other vehicles are going to change lanes in a highway scenario, i.e., the vehicles surrounding the ego vehicle are tracked and a Lane Change Detector (LCD) algorithm detects if a vehicle will change lanes. The inputs to such a driving function are multi-variate time-series data. Recent publications use Recurrent Neural Network (RNN) architectures to classify and predict lane changes [6], [7]. However, neither the validation nor the lacking interpretability of these ML-based driving functions are addressed. In this paper, we introduce an interpretable LCD algorithm to detect when surrounding vehicles change lanes. The algorithm detects a lane change depending on the reconstruction errors of three independent ML-based anomaly detection functions – each trained on a specific driving maneuver. An overview of the algorithm is depicted in Fig. 1. Since the classification is based on explicit decision rules [3, Ch. 4.5], the algorithm is interpretable.

First, we discuss the different approaches to detect anomalies—also known as outliers—in time-series data. An overview of ML methods for anomaly detection can be found in [8]. The Deep Autoencoder (DAE) [9], [10] is a popular anomaly detection method (see, e.g., [8]) because it is trained in an unsupervised manner, i.e., it detects anomalies directly based on the intrinsic properties of the data. A DAE attempts to reconstruct the input signal whilst representing the essential information in a smaller dimensional space, i.e., it aims to minimize the reconstruction error between input and output signals with a smaller dimensional latent space between them. The proposed LCD algorithm is based on DAEs.

In [11], a time-series anomaly detection algorithm is

This research was funded by Audi AG.
Department of Electrical and Computer Engineering, Technical University of Munich, 80333 Munich, Germany; Email: oliver.de-candido@tum.de.

introduced using a Convolutional Neural Network (CNN) with an encoder/decoder architecture, similar to the structure of a DAE. The algorithm is able to detect and classify different types of anomalous signals in streaming time-series data. In [12], an anomaly detection algorithm is introduced to detect abnormal trajectories of road participants at various intersections. The authors train a DAE on sequential data recorded at these intersections. They then classify unseen scenarios as being "normal" or "abnormal" depending on the reconstruction error at the output of the DAE, i.e., if the reconstruction error is larger than a pre-defined threshold, the scenario is classified as an anomaly ("abnormal").

Recently, researchers have employed DAE algorithms to reconstruct, predict, and generate vehicle trajectories in highway scenarios. The authors of [13] introduce an algorithm to predict the future trajectory of vehicles using an RNN-based DAE, which also takes the driver's lane change intention into consideration. They show that this DAE architecture can predict the lateral and longitudinal position of vehicles up to 5 s into the future. In [14], the authors introduce unsupervised ML-based methods to generate lane change trajectories. They show that a DAE algorithm is able to generate new realistic trajectories by varying the latent space representations. This work was extended in [15] using Bézier curves to generate smoother trajectories, but the underlying DAE architecture remains the same.

With these results in mind, we propose a novel LCD algorithm, depicted in Fig. 1, which takes advantage of the anomaly detection performance of DAEs. The algorithm classifies the lane changes using the reconstruction error at the output of the DAE (see, e.g., [12]). To this end, we train three separate DAEs (depicted in the dashed box in Fig. 1): one to reconstruct left lane changes ($f_{\text{left}}$), one to reconstruct right lane changes ($f_{\text{right}}$), and one to reconstruct lane keeping ($f_{\text{keep}}$). Since these DAEs are trained independently on independent datasets, we argue that the anomalies they detect are also independent. By observing the reconstruction error at the output of each DAE, we define a decision rule-set based on pre-determined thresholds to detect an upcoming lane change. By pre-determining these thresholds, an engineer is able to trade-off the classification performance of the detection algorithm with a reliable detection time, i.e., whether the algorithm can reliably detect a lane change earlier or later. Moreover, since the LCD algorithm depends on the independent outputs of three DAEs, we show that the classification is directly interpretable (see Section IV-D). It is well known, see, e.g., [1], [16], that an increase in interpretability of ML-based classifiers comes at the cost of classification performance.

The paper is structured as follows: In Section II, we define the ML problem we consider using DAEs including the real-world dataset used for training the DAEs. In Section III, we introduce the LCD algorithm and the lane change classification algorithms used as references. In Section IV, we provide and discuss various experimental results. In Section V, we conclude the paper with a discussion of the work and possible extensions of it.

| Attribute | Description |
|---|---|
| $v_{\text{lat.}}$ | Lateral velocity |
| $v_{\text{long.}}$ | Longitudinal velocity |
| $a_{\text{lat.}}$ | Lateral acceleration |
| $d_{\text{left}}$ | Distance to the left lane marking |
| $d_{\text{right}}$ | Distance to the right lane marking |

TABLE I: The considered attributes ($\Gamma = 5$).

## II. Problem Formulation

In this section, we introduce the dataset generation and pre-processing steps required by the proposed algorithm. Next, we define the DAE architecture and the problem of detecting anomalies in multi-variate time-series data.

### A. Dataset Generation

As training data for the DAEs we use the realistic driving data summarized in the highD dataset [17]. These data were recorded by flying a drone above the German highway at different locations. They summarize many hours of realistic highway driving data which include thousands of lane change maneuvers. During the first stage of pre-processing we extract the driving maneuvers, i.e., left lane changes, right lane changes, and lane keeping, from the highD dataset; these define the scenarios the DAEs should reconstruct. For the lane keeping data, a scenario is defined where the observed vehicle remains in its lane within all time-steps.

Each DAE is trained using windowed scenarios with the parameter $N_{\text{lc}}$ to indicate the time before the lane change which the DAE should be able to reconstruct, i.e., the number of time-steps before a lane change which should be reconstructed. We create a training dataset for the DAEs as follows: first, a set of scenarios is created

$$\mathcal{D}'_j = \left\{ \tilde{\boldsymbol{X}}^{(1)}, \dots, \tilde{\boldsymbol{X}}^{(M_j)} \right\}, \tag{1}$$

with the dataset label $j =$ left, right, keep and each scenario is defined as

$$\tilde{\boldsymbol{X}} = \left[ \boldsymbol{x}[1], \quad \boldsymbol{x}[2], \quad \dots, \quad \boldsymbol{x}[N_{\text{lc}}] \right] \in \mathbb{R}^{\Gamma \times N_{\text{lc}}}, \tag{2}$$

where the event, e.g., the center of a vehicle crosses the right lane marking, occurs at time-step $N_{\text{lc}}$. At each time-stamp $n = 1, \dots, N_{\text{lc}}$, we summarize the $\Gamma$ attribute signals in a vector $\boldsymbol{x}[n] = [x_1[n], x_2[n], \dots, x_\Gamma[n]]^{\text{T}} \in \mathbb{R}^\Gamma$. The input attributes we consider are summarized in Table I.

Once we have a set of scenarios describing different maneuvers—each with a different number of samples—we further process the data by passing a sliding window over them with a constant window size $W$ [18]. Each scenario, i.e., sample in $\mathcal{D}'_j$, is processed by a sliding window function which generates all possible windows for a scenario, i.e.,

$$\begin{aligned} \text{win}(\tilde{\boldsymbol{X}}; W) = \{ \left[ \boldsymbol{x}[i], \quad \boldsymbol{x}[i+1], \quad \dots, \quad \boldsymbol{x}[i+W-1] \right] : \\ \forall i \in \{1, \dots, N_{\text{lc}} - W + 1\} \} \end{aligned} \tag{3}$$

where the index $i$ indicates the initial time-stamp of each window in a scenario. Thus, the dataset for each driving

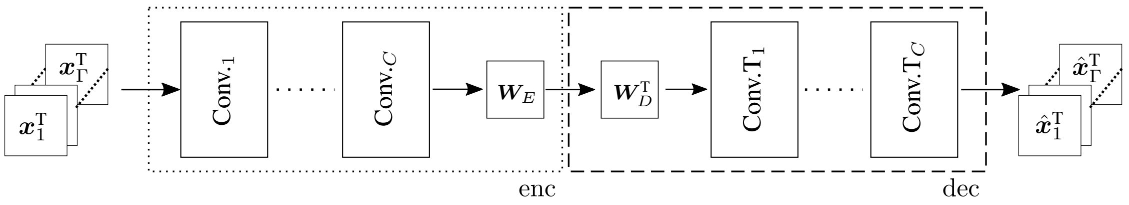


Fig. 2: Abstract DAE architecture.

maneuver can be summarized as

$$\mathcal{D}_j = \left\{ \text{win}\left(\tilde{\boldsymbol{X}}^{(1)}; W\right), \ldots, \text{win}\left(\tilde{\boldsymbol{X}}^{(M_j)}; W\right) \right\}, \quad (4)$$

with $|\mathcal{D}_j| = M_j(N_{\text{lc}} - W + 1)$, the dataset label $j =$ left, right, keep, and each sample, after windowing, has the dimension $\boldsymbol{X} \in \mathbb{R}^{\Gamma \times W}$.

With a dataset defined for each of the three maneuvers, we split them into three disjunct sets for: (i) training the DAE ($\mathcal{D}_{\text{tr},j}$); (ii) determining the threshold values ($\mathcal{D}_{\text{val},j}$); and (iii) testing the detection performance ($\mathcal{D}_{\text{test},j}$). We separate the dataset into a 70%, 10% and 20% split of the total data $|\mathcal{D}_j|$, respectively.

## B. Deep Autoencoders

Throughout this work, we train DAEs with 1-D convolutional layers; convolutional based DAEs were first introduced in [10]. It has been shown that these layers can be employed to extract discriminative features in multi-variate time-series data (see, e.g., [19], [20]).

The general architecture of a DAE with 1-D convolutional layers is depicted in Fig. 2. On the left hand side, we observe the $\Gamma$ input channels of the multi-variate time-series datum which are first passed through an encoder function $\text{enc} : \mathbb{R}^{\Gamma \times W} \to \mathbb{R}^p$ (dotted box), and then a decoder function $\text{dec} : \mathbb{R}^p \to \mathbb{R}^{\Gamma \times W}$ (dashed box). At the output, the decoder attempts to reconstruct each input signal, i.e., $\boldsymbol{x}_\gamma^{\text{T}} \approx \hat{\boldsymbol{x}}_\gamma^{\text{T}}$. The encoder and the decoder are designed symmetrically, i.e., they have the same number of convolutional blocks and transposed convolutional blocks, respectively. Before the signal is propagated into the latent space of dimension $\boldsymbol{z} \in \mathbb{R}^p$, i.e., the space between the encoder and the decoder, a fully-connected layer is employed. The first layer in the decoder is also a fully-connected layer.

The overall structure of a DAE can be summarized as

$$f_j(\boldsymbol{X}) = \text{dec}(\text{enc}(\boldsymbol{X}; \mathbb{E}); \mathbb{D}), \quad (5)$$

with the encoder parameters $\mathbb{E}$ and the decoder parameters $\mathbb{D}$. The index $j =$ right, left, keep indicates which DAE is trained, i.e., which training dataset is used.

We use the Mean Squared Error (MSE) loss function to train the DAEs, i.e.,

$$\mathcal{L}(\mathcal{D}_{\text{tr},j}; \mathbb{E}, \mathbb{D}) = \frac{1}{M_{\text{tr},j} W \Gamma} \sum_{i=1}^{M_{\text{tr},j}} \left\| \boldsymbol{X}^{(i)} - f_j(\boldsymbol{X}^{(i)}) \right\|_F^2, \quad (6)$$

with the Frobenius norm $\|\cdot\|_F$ and the DAE for maneuver $j =$ left, right, keep, as defined in (5). Since the three DAEs are trained with the loss defined in (6) on independent datasets the resulting DAE function will embed and reconstruct an unseen input differently. Thus, the reconstruction errors of the three DAEs are independent.

## C. Anomaly Detection

A main benefit of training a DAE on sliding window multi-variate time-series data is the ability to detect anomalous signals (see, e.g., [8], [12]). Since the DAE can reconstruct data which are from the same distribution as the training set, when a datum from a different distribution is input, the reconstruction error at the output will be larger than for a datum from the same distribution.

In our case, three DAEs are trained on left lane change, right lane change and lane keeping maneuvers, contained in $\mathcal{D}_{\text{tr, left}}$, $\mathcal{D}_{\text{tr, right}}$ and $\mathcal{D}_{\text{tr, keep}}$, respectively. Therefore, we expect a higher reconstruction error at the output of the DAE when a datum from a different dataset, i.e., a different driving maneuver, is input. We take advantage of this fact in the proposed LCD algorithm.

## III. Lane Change Detection Algorithm

In this section, we describe the proposed LCD algorithm and introduce the parameters which can be optimized with respect to the engineering requirements. The key ideas behind the algorithm were introduced in Section II, i.e., the windowing of a driving maneuver, the DAE architecture, and the anomaly detection ability of DAEs.

In the first stage of the algorithm, three DAEs, $f_{\text{left}}$, $f_{\text{right}}$, and $f_{\text{keep}}$, should be trained on representative datasets containing different lane change maneuvers. Once the DAEs have been trained, the weights are not changed, and the DAEs are only used to reconstruct the input. Since the DAEs are trained independently, the anomalous signals each detects will be independent. As the decision only relies on the reconstruction errors of these independent DAEs, the decision of the LCD algorithm is fully interpretable.

The LCD algorithm is depicted in Algorithm 1. For an arbitrary input, $\bar{\boldsymbol{X}}$, we first calculate the reconstruction error of each DAE (Lines 2, 3 and 4). Next, we calculate the difference $\delta_{\text{keep}}$ between the current reconstruction error $\varepsilon_{\text{keep}}$ and the reconstruction error from the previous time-stamp $\varepsilon'_{\text{keep}}$. By tracking the change in reconstruction error of $f_{\text{keep}}$, we track the derivative of the reconstruction error. This improves the performance of the LCD algorithm; an improvement in performance was not observed when considering $\delta_{\text{left}}$ or $\delta_{\text{right}}$. With the given threshold values $\tau_{\text{left}}$, $\tau_{\text{right}}$, $\tau_{\text{keep}}$

and $\tau_\delta$ (see Section III-A), the algorithm estimates which class the current window belongs to using explicit decision rules [3, Ch. 4.5]. This highlights the direct interpretability of the LCD algorithm. A left lane change is detected if $f_{\text{right}}$ cannot reconstruct the current datum, and either $f_{\text{keep}}$ cannot reconstruct it as well or the difference $\delta_{\text{keep}}$ is large, simultaneously, $f_{\text{left}}$ can reconstruct the datum (see Line 6, where $\wedge$ and $\vee$ represent a logical "and" and a logical "or" operator, respectively). A similar rule-set is used to detect right lane changes (see Line 8). If the current datum is not a lane change, the label at the output is $c = 0$ (lane keeping).

**Algorithm 1** Lane Change Detector (LCD) Algorithm

```
 1: Inputs: Multi-variate time-series datum: $\bar{\boldsymbol{X}} \in \mathbb{R}^{\Gamma \times W}$,
    reconstruction error from previous time-stamp: $\varepsilon'_{\text{keep}}$
 2: $\varepsilon_{\text{left}} \leftarrow \|\bar{\boldsymbol{X}} - f_{\text{left}}(\bar{\boldsymbol{X}})\|_F^2$
 3: $\varepsilon_{\text{right}} \leftarrow \|\bar{\boldsymbol{X}} - f_{\text{right}}(\bar{\boldsymbol{X}})\|_F^2$
 4: $\varepsilon_{\text{keep}} \leftarrow \|\bar{\boldsymbol{X}} - f_{\text{keep}}(\bar{\boldsymbol{X}})\|_F^2$
 5: $\delta_{\text{keep}} \leftarrow \varepsilon_{\text{keep}} - \varepsilon'_{\text{keep}}$
 6: if $(\varepsilon_{\text{keep}} \geq \tau_{\text{keep}} \vee \delta_{\text{keep}} \geq \tau_\delta) \wedge (\varepsilon_{\text{right}} \geq \tau_{\text{right}}) \wedge (\varepsilon_{\text{left}} < \tau_{\text{left}})$
    then
 7:     $c \leftarrow -1$ {Left Lane Change}
 8: else if $(\varepsilon_{\text{keep}} \geq \tau_{\text{keep}} \vee \delta_{\text{keep}} \geq \tau_\delta) \wedge (\varepsilon_{\text{left}} \geq \tau_{\text{left}}) \wedge$
    $(\varepsilon_{\text{right}} < \tau_{\text{right}})$ then
 9:     $c \leftarrow +1$ {Right Lane Change}
10: else
11:     $c \leftarrow 0$ {Lane Keeping}
12: end if
13: return $c, \varepsilon_{\text{keep}}$
```

## A. Performance Trade-off: Threshold Determination

An important step in the design of the LCD algorithm is to determine the threshold values for the different DAEs. These thresholds—as seen in Algorithm 1—not only determine how well the algorithm can classify each maneuver, but also how early a lane change is reliably detected.

On the one hand, we use the macro-averaged $F_1$ score [21] as the classification performance measure we aim to maximize. The precision score and the recall score are defined for each class $c$ as

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \ \text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}, \quad (7)$$

with the true positives ($\text{TP}_c$), false positives ($\text{FP}_c$) and false negatives ($\text{FN}_c$) of a given class $c$. The $F_{1,c}$ score for that class is calculated as the harmonic mean of the precision and recall scores, i.e.,

$$F_{1,c} = 2\frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}. \quad (8)$$

The macro-averaged $F_1$ score is the arithmetic mean of the $F_{1,c}$ scores per class, i.e., $F_1 = \sum_c F_{1,c}/C$ with the total number of classes $C$. We also consider the macro-averaged precision and recall scores.

To this end, we create an aggregated dataset out of the validation datasets introduced in Section II-A. We label each sample with a label corresponding to the driving maneuver, e.g., all samples in $\mathcal{D}_{\text{val, left}}$ are given the label $c = -1$, all samples in $\mathcal{D}_{\text{val, right}}$ are given the label $c = +1$, and all samples in $\mathcal{D}_{\text{val, keep}}$ are given the label $c = 0$. Thus, we define the validation dataset as

$$\mathcal{D}_{\text{val}} = \left\{ \left(\boldsymbol{X}^{(i)}, c^{(i)}\right) \right\}_{i=1}^{M_{\text{val}}}, \quad (9)$$

where the samples are taken from $\mathcal{D}_{\text{val, left}}, \mathcal{D}_{\text{val, right}}$ and $\mathcal{D}_{\text{val, keep}}$. The total number of samples is $M_{\text{val}} = |\mathcal{D}_{\text{val, left}}| + |\mathcal{D}_{\text{val, right}}| + |\mathcal{D}_{\text{val, keep}}|$.

On the other hand, we want the LCD algorithm to reliably detect a lane change as early as possible before the lane change actually occurs. Thus, we define a reliable detection time as Def. 1.

**Definition 1.** *A detection is considered reliable if the prediction does not change in the time interval between the first prediction and the actual lane change.*

With the validation datasets, we search for the threshold values $\boldsymbol{\tau} = [\tau_{\text{left}}, \tau_{\text{right}}, \tau_{\text{keep}}, \tau_\delta]^{\text{T}} \in \mathbb{R}^4$ which maximize the $F_1$ score whilst maintaining a reliable early detection.

We further define the set of reconstruction errors of each DAE on the validation dataset as

$$\mathcal{S}_j = \left\{ \|\boldsymbol{X} - f_j(\boldsymbol{X})\|_F^2 \ : \ \forall \boldsymbol{X} \in \mathcal{D}_{\text{val},j} \right\}, \quad (10)$$

with the trained DAEs $f_j$, the validation dataset for each maneuver $\mathcal{D}_{\text{val},j}$, and $j$ = left, right, keep. Now, we can exhaustively search for the optimal thresholds over the range $\tau_j \in (0, \max\{\mathcal{S}_j\}]$ which is the maximum reconstruction error in the set $\mathcal{S}_j$ as defined in (10).

Additionally, a method to estimate an appropriate anomaly threshold is to estimate the mean and standard deviation of the reconstruction error of each DAE on the validation dataset. This is similar to the method employed in [12]. To this end, we can set each threshold as

$$\tau_j = \text{mean}\{\mathcal{S}_j\} + 3 \times \text{std}\{\mathcal{S}_j\}, \quad (11)$$

with the mean and the standard deviation of the reconstruction error values on each validation dataset as defined in (10). The threshold $\tau_\delta$ can be determined as one standard deviation smaller than the mean difference in lane keep reconstruction errors using the validation datasets of lane changes.

## B. Reference Algorithms

We use two state-of-the-art lane change classification algorithms and a CNN-based algorithm as reference methods. Both of the state-of-the-art classification algorithms use RNN architectures which can handle the multi-variate time-series data. The authors of [6], first introduced an architecture using Long Term Short Term Memory (LSTM) cells [22], and the authors of [7] extended the architecture to use Gated Recurrent Unit (GRU) cells [23]. These both show good lane change classification results, however, neither the problem of interpretability nor the challenge of validation are explicitly mentioned. These will be referred to as LSTM [6] and GRU [7], respectively.

| Network | Layer | Input | Output | Kernel | Stride |
|---|---|---|---|---|---|
| Encoder | Conv1D | 5 Chs | 10 Chs | $1 \times 3$ | 2 |
| | Conv1D | 10 Chs | 20 Chs | $1 \times 3$ | 2 |
| | Conv1D | 20 Chs | 30 Chs | $1 \times 3$ | 2 |
| | Linear | 60 Ns | 5 Ns | N/A | N/A |
| Decoder | Linear | 5 Ns | 60 Ns | N/A | N/A |
| | ConvT1D | 30 Chs | 20 Chs | $1 \times 3$ | 2 |
| | ConvT1D | 20 Chs | 10 Chs | $1 \times 3$ | 2 |
| | ConvT1D | 10 Chs | 5 Chs | $1 \times 3$ | 2 |

TABLE II: The DAE architecture for the LCD algorithm. The input/output columns denote the number of channels (Chs) for the convolution layers and number of neurons (Ns) for the linear layers.

Additionally, due to the recent success of CNN architectures in multi-variate time-series classification tasks (see, e.g., [19], [20]), we compare the interpretable LCD algorithm with a CNN-based classifier. It also lacks direct interpretability of its classifications.

## IV. Experimental Results

In this section, we discuss the experimental setup and results we achieved with the LCD algorithm introduced in Section III compared with the reference algorithms. We show how lane changes are detected in an interpretable manner and how the threshold parameters can be chosen by an engineer.

### A. Experimental Setup

We use the same architecture for each DAE, with a symmetric encoder and decoder design. The parameters are summarized in Table II. The latent space has a dimension $\boldsymbol{z} \in \mathbb{R}^5$.

Since the input attributes are real-valued, i.e., $\boldsymbol{x}[n] \in \mathbb{R}^\Gamma$, we use the Tanhshrink activation function, defined as

$$\operatorname{tanhshrink}(\boldsymbol{a}) = \boldsymbol{a} - \tanh(\boldsymbol{a}), \tag{12}$$

which is applied element-wise on an input $\boldsymbol{a} \in \mathbb{R}^n$.

We train each DAE using the training data $\mathcal{D}_{\text{tr},j}$ for the respective driving maneuver with the MSE loss defined in (6). We assume the lane changes occur at time-stamp $N_{\text{lc}} = 100$, i.e., the scenarios before windowing are of dimension $\tilde{\boldsymbol{X}} \in \mathbb{R}^{5\times 100}$ (*cf.* (2)). This means the DAEs attempt to reconstruct 4 s before the lane change – this is motivated by the results from [13]. Moreover, we use a window of length $W = 25$ time-stamps which corresponds to subsequences of the time-series data of length 1 s. Thus, the inputs to the DAEs are of dimension $\boldsymbol{X} \in \mathbb{R}^{5\times 25}$ (*cf.* (4)).

We train each DAE for 200 epochs using the Adam optimizer [24] with mini-batches of size 200 and a learning rate of $\alpha = 0.0001$.

To train the reference algorithms, we use the same training data which we trained the DAEs on, i.e., $\mathcal{D}_{\text{tr, left}}, \mathcal{D}_{\text{tr, right}}$ and $\mathcal{D}_{\text{tr, keep}}$. Furthermore, we label each sample using the same principle as introduced in Sub-section III-A, i.e., each left lane change maneuver is labelled with $c = -1$, each right lane change maneuver is labelled with $c = +1$, and each lane keep maneuver is labelled with $c = 0$. Given this training set, we can train the networks using the standard cross-entropy classification loss [25, Ch. 4]. The reference algorithms are trained with the same hyper-parameters as the DAEs, i.e., the same number of epochs, mini-batch size and learning rate. The RNN architectures were taken from [6] and [7]. For the CNN architecture, we take the encoder design from Table II and change the output to 3 neurons – one for each class.

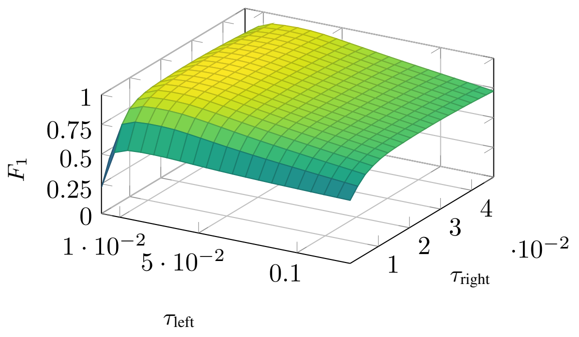
### B. Threshold Determination

First, we discuss the threshold determination which is required for the LCD algorithm. As the LCD algorithm is designed to detect lane changes in an interpretable manner, we show that an engineer is able to choose the threshold values such that the desired performance is achieved. First, we calculate the threshold values using the means and the standard deviations on the validation datasets as introduced in (11). This results in the threshold values of $\tau_{\text{keep, std}} = 0.015$, $\tau_{\text{left, std}} = 0.027$, $\tau_{\text{right, std}} = 0.021$, and $\tau_{\delta,\text{std}} = 2.97{\times}10^{-4}$. We denote the algorithm with these threshold values as $\text{LCD}_{\text{std}}$.

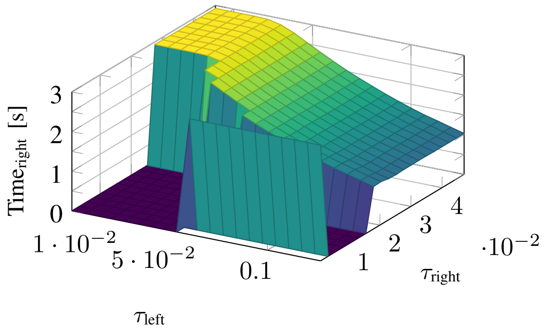
We can also determine the threshold values depending on the desired performance of the LCD algorithm. Thus, we fix the keep thresholds $\tau_{\text{keep, det}} = \tau_{\text{keep, std}}$ and $\tau_{\delta,\text{det}} = \tau_{\delta,\text{std}}$. Now, we can search for the left and right thresholds which determine the desired function performance. To this end, we plot the macro-averaged $F_1$ score (*cf.* (8)), the average reliable detection time for left and for right lane changes (*cf.* Def. 1) on the validation dataset in Fig. 3. We consider the average reliable detection time only when at least 93% of the validation data is reliably detected. In Fig. 3a, we see that a threshold value $\tau_{\text{left}} \in (0.021, 0.034)$ and $\tau_{\text{right}} \in (0.013, 0.035)$ achieves an $F_1$ score of over 95%. For a reliable detection time of 2.9 s for right lane changes we observe that $\tau_{\text{left}} \in (0.007, 0.034)$ and $\tau_{\text{right}} \in (0.028, 0.047)$ in Fig. 3b. An average reliable detection time of at least 2.85 s for left lane changes is achieved for $\tau_{\text{left}} \geq 0.027$ and $\tau_{\text{right}} \leq 0.28$ as seen in Fig. 3c. With these results in mind, we choose the left and the right threshold values to be $\tau_{\text{left, det}} = 0.027$ and $\tau_{\text{right, det}} = 0.028$, respectively. We denote the algorithm with these threshold values as $\text{LCD}_{\text{det}}$.

### C. Lane Change Detection Results

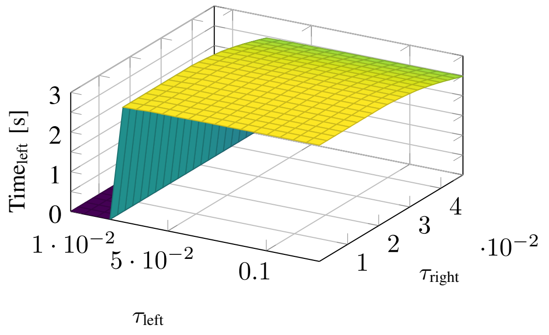
Now, we discuss the classification results achieved by the different LCD algorithms on the test dataset. We observe in Table IIIa that the purely ML-based algorithms achieve a better classification performance than the interpretable LCD algorithms. The CNN algorithm outperforms the RNN-based algorithms on this task. Since the encoders of the DAEs use the same CNN architecture, the classification results support the argument that a CNN is able to extract discriminative features. The interpretable LCD algorithm achieves a classification performance to within 3.5% of the black-box ML-based methods; the average precision is only 2.5% away. We observe that the $\text{LCD}_{\text{det}}$ algorithm achieves a slightly better performance compared with the $\text{LCD}_{\text{std}}$ algorithm, which indicates the benefit of determining the threshold values manually. These classification results show that the



(a) $F_1$ score.



(b) Average reliable detection time for right lane changes.



(c) Average reliable detection time for left lane changes.

Fig. 3: Threshold choice showing the trade-off between high $F_1$ score and a reliable early detection.

| Alg. | Acc. [%] | $F_1$ [%] | Precision [%] | Recall [%] |
|---|---|---|---|---|
| LSTM [6] | 99.5 | 99.6 | 99.6 | 99.6 |
| GRU [7] | 99.5 | 99.5 | 99.5 | 99.6 |
| CNN | **99.7** | **99.8** | **99.7** | **99.8** |
| $\text{LCD}_{\text{std}}$ | 96.4 | 96.5 | 97.2 | 96.0 |
| $\text{LCD}_{\text{det}}$ | 96.5 | 96.6 | 97.3 | 96.0 |

(a) Lane change detection classification results.

| Alg. | $\text{Time}_{\text{left}}$[s] | Rel. [%] | $\text{Time}_{\text{right}}$[s] | Rel. [%] |
|---|---|---|---|---|
| LSTM [6] | 3.03 | **99.8** | 3.03 | 98.4 |
| GRU [7] | 3.03 | 98.6 | **3.04** | 98.8 |
| CNN | **3.04** | **99.8** | **3.04** | **99.8** |
| $\text{LCD}_{\text{std}}$ | 2.93 | 94.1 | 2.93 | 92.6 |
| $\text{LCD}_{\text{det}}$ | 2.90 | 94.3 | 2.93 | 94.8 |

(b) The average reliable detection time results, including a percentage of reliable (Rel.) detections on the test dataset, for left and for right lane changes.

TABLE III: Classification and reliable detection time results on the test data with $N_{\text{lc}} = 100$.

proposed LCD algorithm is able to classify the different driving maneuvers whilst maintaining interpretability.

In Table IIIb, we observe the average reliable detection time results for the different algorithms. Again, the LCD algorithms almost achieve the same performance as the black-box ML-based classification algorithms. On average, the algorithm can reliably predict a lane change $2.93$ s before lane changes, only $0.1$ s away from the RNN-based methods. Furthermore, we observe that the $\text{LCD}_{\text{det}}$ detections are more reliable than the $\text{LCD}_{\text{std}}$ algorithm, i.e., in at least $94.3\%$ of the test data the algorithm output a reliable detection (see Def. 1) for both types of lane changes. Recall, we chose the threshold values to achieve a reliability of at least $93\%$ on the validation data.

Overall, the simulation results indicate that the proposed LCD algorithm can reliably detect lane changes; however, it shows a slight performance degradation compared to state-of-the-art ML algorithms. This is the price of the interpretability of the classification (see, e.g., [1], [16]).

### D. Interpretability of the LCD Algorithm

In the previous sub-section, we saw that the LCD algorithm is able to classify different driving maneuvers almost as well as state-of-the-art ML algorithms. The reasons why ML-based algorithms make certain classifications is opaque and not directly interpretable. Now, we demonstrate how the proposed LCD algorithm is interpretable. To this end, we take three scenarios—one for each driving maneuver—from the test dataset and assume $N_{\text{lc}} = 200$. Then, we pass each scenario through the window function (see (3)), and classify each sample sequentially.

The results of passing each windowed sample through the classifiers are depicted in Figure 4. The first driving maneuver—a vehicle changing lanes to the right—is depicted between $1$ s and $8$ s. We see that the $\text{LCD}_{\text{det}}$ algorithm outputs the label $c_{\text{LCD}_{\text{det}}} = +1$, only once all conditions in Line 8 of Algorithm 1 are fulfilled, i.e., $f_{\text{left}}$ detects an anomalous signal, the difference between reconstructions of the DAE for lane keeping is large, and $f_{\text{right}}$ is able to reconstruct the signal. The classification occurs 3.6 s before the right lane change for the given threshold values. Moreover, the reconstruction error at the output of $f_{\text{right}}$ decreases and the reconstruction errors of $f_{\text{left}}$ and $f_{\text{keep}}$ increase after the lane change is detected. We observe in the top plot that both of the RNN-based methods show an erroneous label ($c_{\text{GRU [7]}} = c_{\text{LSTM [6]}} = 0$) after initially classifying the windows as a lane change to the right. This would not be considered a reliable detection as per Def. 1. The reason for this change in decision is not obvious nor directly explainable by the RNN methods.

Between $8$ s and $15$ s, all four classifiers correctly detect that the vehicle remains in its lane ($c = 0$). From the recon-

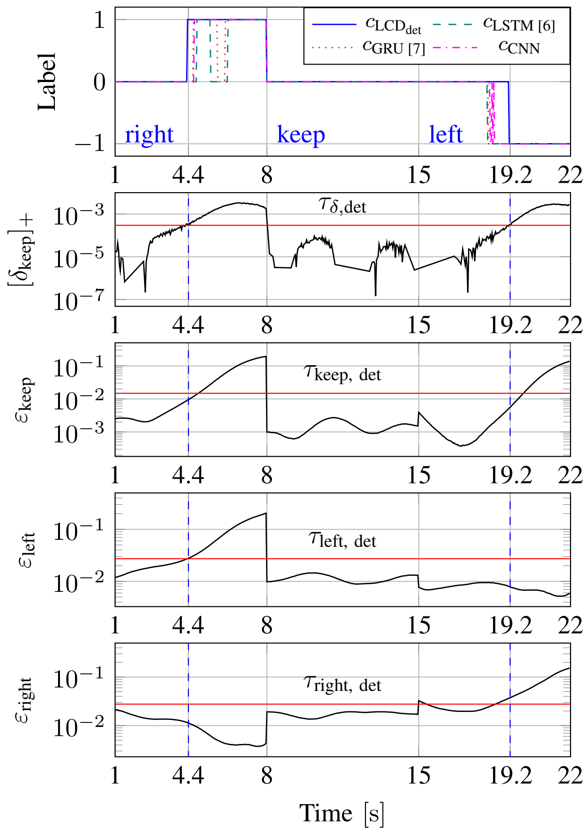


Fig. 4: Classification output for different driving maneuvers, illustrating the interpretability of the LCD algorithm.

struction error plots, we observe that all three DAEs were able to reconstruct the samples. Finally, a scenario where the vehicle changes lanes to the left is depicted between 15 s and 22 s. Again, by observing the reconstruction errors, we can directly see why the LCD algorithm classified the sample as a left lane change. It is interesting to note that the DAE for right lane change maneuvers, $f_{\mathrm{right}}$, detected an anomalous signal earlier than the DAE for lane keeping. This leads to the lane change only being correctly classified 2.8 s before it occurred.

These results not only highlight the interpretability of the LCD algorithm, but they also show that the reconstruction error at the output of each DAE is independent for different inputs. Moreover, we demonstrate how one can easily understand the detections made by the LCD algorithm.

### E. Performance Trade-off: Early vs. Reliable Detections

The results thus far highlight the interpretability of the LCD algorithm, and show that the performance is comparable to purely ML-based methods. Now, we investigate the performance trade-off between early and reliable detections by varying the threshold $\tau_\delta$. We use a test dataset with $N_{\mathrm{lc}} = 200$ to investigate the generalizability of the algorithm.

The LSTM [6] method achieves the best performance with an average detection time of $\mathrm{Time}_{\mathrm{left}}$ of 3.70 s and $\mathrm{Time}_{\mathrm{right}}$ of 3.49 s with a reliability of 91.9% and 93.6%, respectively. This performance cannot directly be improved.

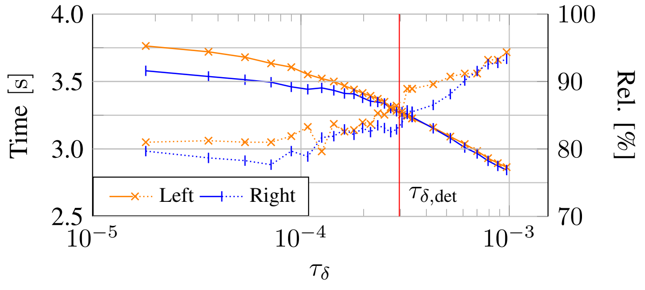


Fig. 5: Average reliable detection time (solid lines) and percentage of reliable detections (dotted lines) plotted against the threshold $\tau_\delta$. Left lane changes indicated by: $\times$; right lane changes indicated by: $|$.

In Fig. 5, we plot the trade-off between an early average reliable detection time and the reliability estimate depending on the threshold value $\tau_\delta$. We observe that with a small threshold value, we achieve an average reliable detection time of 3.76 s and 3.58 s for left and right lane changes, respectively. However, this comes at the cost of less reliable detections. On the other hand, if we make $\tau_\delta$ large enough, we observe a reliability estimation of 94.3% and 93.4%, which comes at the cost of a smaller average detection time. Furthermore, the value $\tau_{\delta,\mathrm{det}}$, which we use in our algorithms, is near the intersection point of the curves. These results not only emphasize the benefit of a fully interpretable LCD algorithm, but they also show how the threshold values can be chosen to achieve a desired performance.

## V. Conclusion and Outlook

Motivated by the ability of DAEs to detect anomalous signals in multi-variate time-series data, we propose a LCD algorithm based on the reconstruction error of three independent DAEs. We demonstrate how an engineer can choose appropriate threshold values for the LCD algorithm. The performance capability of the LCD algorithm is investigated on realistic highway driving data. We show that the interpretable algorithm almost performs as well as black-box ML-based methods. Moreover, we highlight the inherent interpretability of the novel LCD algorithm.

This work demonstrates the feasibility of building an interpretable classifier out of black-box ML algorithms. The method can be extended by further optimizing the anomaly detection, i.e., not only using the output reconstruction error but also taking the activations in the hidden-layers into account when detecting anomalous signals (see, e.g., [26]). Additionally, one can investigate the influence $N_{\mathrm{lc}}$ has on the performance of the algorithms. Moreover, the interpretability can be extended if one were to consider the reconstruction error of each signal individually. Thus, potentially enabling the LCD algorithm to also explain why a lane change is imminent, e.g., the longitudinal acceleration is typical for a left lane change and not for a right lane change.

## References

[1] DARPA, "Explainable Artificial Intelligence (XAI)," Broad Agency Announcement: DARPA-BAA-16-53, Aug. 2016. [Online]. Available: https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf

[2] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer Nature, 2019, vol. 11700.

[3] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[4] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *Int. conf. on Computer Vision*. IEEE, 2017, pp. 2942–2950.

[5] W. Zeng *et al.*, "End-to-end interpretable neural motion planner," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 8660–8669.

[6] H. Q. Dang, J. Fürnkranz, A. Biedermann, and M. Hoepfl, "Time-to-lane-change prediction with deep learning," in *Intell. Transp. Syst. Conf. (ITSC)*. IEEE, 2017.

[7] Z. Yan *et al.*, "Time to lane change and completion prediction based on Gated Recurrent Unit Network," in *Intell. Vehicles Symp. (IV)*. IEEE, 2019, pp. 102–107.

[8] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019. [Online]. Available: http://arxiv.org/abs/1901.03407

[9] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[10] J. Masci, U. Meier, D. Cirean, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Lecture Notes in Comput. Sci.*, vol. 6791 LNCS, 2011, pp. 52–59.

[11] T. Wen and R. Keyes, "Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning," 2019. [Online]. Available: http://arxiv.org/abs/1905.13628

[12] P. R. Roy and G. A. Bilodeau, "Road user abnormal trajectory detection using a deep autoencoder," in *Lecture Notes in Comput. Sci.*, vol. 11241 LNCS, 2018, pp. 748–757.

[13] X. Feng, Z. Cen, J. Hu, and Y. Zhang, "Vehicle Trajectory Prediction Using Intention-based Conditional Variational Autoencoder," in *Intell. Transp. Syst. Conf. (ITSC)*. IEEE, 2019, pp. 3514–3519.

[14] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein, "Data-Driven Maneuver Modeling using Generative Adversarial Networks and Variational Autoencoders for Safety Validation of Highly Automated Vehicles," in *Intell. Transp. Syst. Conf. (ITSC)*. IEEE, 2018.

[15] R. Krajewski, T. Moers, A. Meister, and L. Eckstein, "BézierVAE: Improved Trajectory Modeling using Variational Autoencoders for the Safety Validation of Highly Automated Vehicles," in *Intell. Transp. Syst. Conf. (ITSC)*. IEEE, 2019.

[16] D. Bersimas, P. Jaillet, A. Delarue, and S. Martin, "The price of interpretability," 2019. [Online]. Available: http://arxiv.org/abs/1907.03419

[17] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Intell. Transp. Syst. Conf. (ITSC)*. IEEE, 2018.

[18] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Int. Conf. on Data Mining (ICDM)*. IEEE, 2001, pp. 289–296.

[19] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Int. Joint Conf. on Neural Netw.* IEEE, 2017, pp. 1578–1585.

[20] O. De Candido *et al.*, "Towards Feature Validation in Time to Lane Change Classification using Deep Neural Networks," in *Intell. Transp. Syst. Conf. (ITSC)*. IEEE, 2020, pp. 1697–1704.

[21] C. J. Van Rijsbergen, *Information Retrieval*. Butterworths, 1979.

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

[23] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: EncoderDecoder Approaches," 2015. [Online]. Available: http://arxiv.org/abs/1409.1259

[24] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. on Learn. Representations (ICLR)*, 2015.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[26] K. Kim *et al.*, "RAPP: Novelty Detection with Reconstruction Along Projection Pathway," in *Int. Conf. on Learn. Representations (ICLR)*, 2020.