applied mechanics

Article

Generation of Realistic Cut-In Maneuvers to Support Safety Assessment of Advanced Driver Assistance Systems

Zafer Kayatas [1,*], Dieter Bestle [2], Pascal Bestle [1] and Robin Reick [1]

[1] Mercedes-Benz AG, Kolumbusstr. 19+21, 71063 Sindelfingen, Germany
[2] Department of Engineering Mechanics and Vehicle Dynamics, Brandenburg University of Technology Cottbus-Senftenberg, Siemens-Halske-Ring 14, 03046 Cottbus, Germany; bestle@b-tu.de
* Correspondence: zafer.kayatas@mercedes-benz.com

Abstract: Advanced Driver Assistance Systems (ADASs) attract constantly growing attention from academics and industry as more and more vehicles are equipped with such technology. Level-3 ADASs, like the DRIVE PILOT from Mercedes-Benz AG, are expected to appear more and more on the market in the next few years. However, automated driving raises new challenges for the system validation required for series approval. The replacement of a human driver as control instance expands the range of variants to be validated and verified. The scenario-based validation approach meets these challenges by simulating only specific safety-critical driving scenarios using software-in-the-loop simulation. According to the current state of the art, various safety-relevant driving scenarios are parameterized as idealized maneuvers which, however, requires a great modeling effort, and at the same time, such simplifications may bias the safety assessment. Therefore, a novel approach using artificial intelligence methods is taken here to generate more realistic driving scenarios. Namely, a generative model based on a variational autoencoder is trained with real-world data and then used to generate trajectories for a specific driving maneuver. Through a comprehensive analysis of the synthetic trajectories, it becomes clear that the generative model can learn and replicate relevant properties of real driving data as well as their probabilistics much better than the mathematical models used so far. Furthermore, it is proven that both the statistical properties and the time characteristics are almost equal to those of the input data.

Keywords: advanced driver assistance system; real traffic situation; cut-in maneuver; neural networks; variational autoencoder; generative modeling; machine learning; synthetic data; software-in-the-loop simulation

check for updates

Citation: Kayatas, Z.; Bestle, D.; Bestle, P.; Reick, R. Generation of Realistic Cut-In Maneuvers to Support Safety Assessment of Advanced Driver Assistance Systems. *Appl. Mech.* **2023**, *4*, 1066–1077. https://doi.org/10.3390/applmech4040054

Received: 16 August 2023
Revised: 18 September 2023
Accepted: 25 September 2023
Published: 28 September 2023

1. Introduction

The validation of Advanced Driver Assistance Systems (ADASs) usually consists of real-world driving tests. With the increasing complexity of ADASs and the resulting increased test effort, these tests are too costly from an economic point of view [1]. More efficient approaches to safeguarding highly automated vehicles are provided by the German research project PEGASUS [2]. In particular, the field of simulative validation offers automated and efficient methods. Especially, the scenario-based approach, where only specific safety-critical driving scenarios are simulated, reduces the test effort to a minimum [3–5].

In such an approach, a simulation tool emulates the real ECU code with the automated driving function to perform software-in-the-loop simulation. The inputs for the simulated control device are generated by a simulation environment, including vehicle models and sensor models, as well as data from other ECUs installed in the system vehicle, also called ego vehicles. To ensure valid input data, the ego vehicle interacts with other traffic participants (road objects) in a virtual environment. The dynamics of the road objects, which ideally follow mathematical models depending on the described driving situations, are recorded by sensor models. Thus, a virtual world that is close to reality is processed and captured by the vehicle model.

Usually, the procedure is based on an ideal mathematical modeling of driving scenarios using physical parameters. For example, the cut-in maneuver in Figure 1a may be parameterized by an initial lateral displacement d_c^0 of the cut-in vehicle (blue) with reference to the ego vehicle (red), headway time t_h^*, and velocity v_c^* when entering the ego lane and lateral distance d_c^1 at the end of the maneuver; see also Figure 1b. Up to 20 parameters are used to describe such a cut-in maneuver and need to be identified for observed instances from field measurements [6].

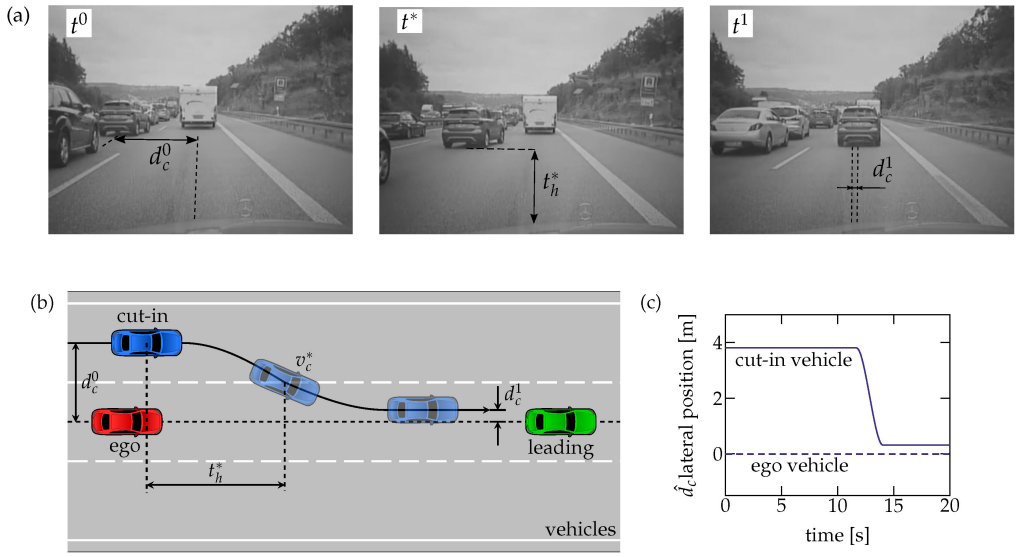

Figure 1. Cut-in scenario: (**a**) image sequence, (**b**) idealized parameterized maneuver, and (**c**) idealized example trajectory of lateral displacement with reference to ego lane.

A specific choice of these parameter values will then result in a specific trajectory representing a simplified real maneuver, as shown in Figure 1c, which, however, is generally not able to represent all aspects of real traffic situations. Especially, unusual trajectories will appear, which can approximate observed trajectories only in a crude way, resulting in biased safety estimates. For example, Figure 2 shows over 8000 real cut-in trajectories measured from real drives of a vehicle fleet. High-intensity sections indicate areas where data points occur more frequently. The majority of the maneuvers have an initial lateral offset of 3 to 4 m from the center of the lane of the ego vehicle before changing lanes, which corresponds approximately to the average lane width of German motorways. After a lane change, the cut-in vehicles are often located in the middle of the ego lane. The highlighted curve picks an arbitrary example trajectory, clearly demonstrating the large difference between the idealized lateral position behavior in Figure 1c and real maneuvers. This demonstrates the need for more realistic approximations.

For safety assessment of ADASs, the generation of realistic maneuvers is not enough, their statistical properties also need to be retained. In the actual approach, the parameters of the idealized maneuvers (Figure 1) are identified for a set of real measured drives on German highways (Figure 2) and evaluated statistically [6], as illustrated in Figure 3a. The extracted probability densities are then used to generate a sample of parameter sets of any required size by Monte Carlo sampling [7], reconstructing corresponding idealized maneuvers as time series and feeding a software-in-the-loop simulation model to evaluate the criticality of these maneuvers. The obtained criticalities can finally be analyzed to perform a safety assessment of the investigated ADAS software which, however, is only as reliable as the underlying maneuvers representing the real traffic situations.

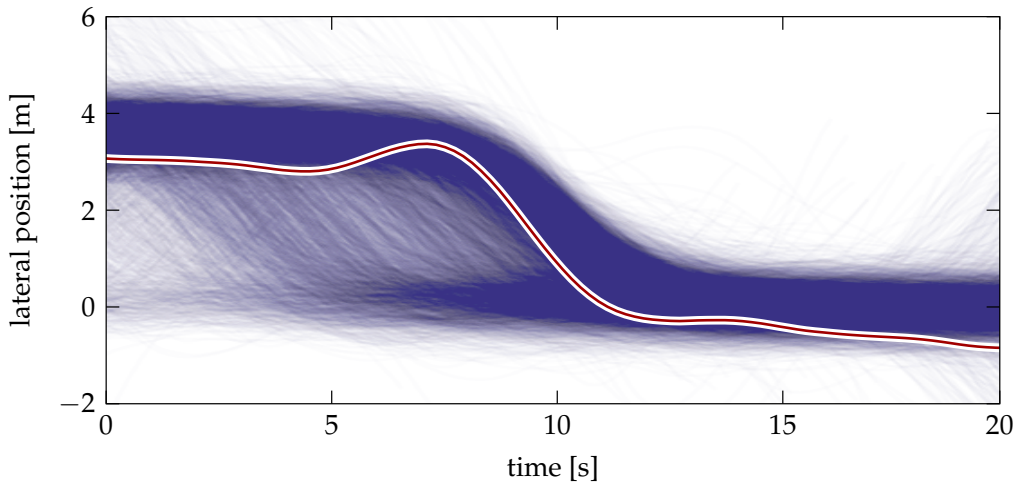

Figure 2. Variety of measured cut-in maneuvers with one real example trajectory for lateral displacement being highlighted.

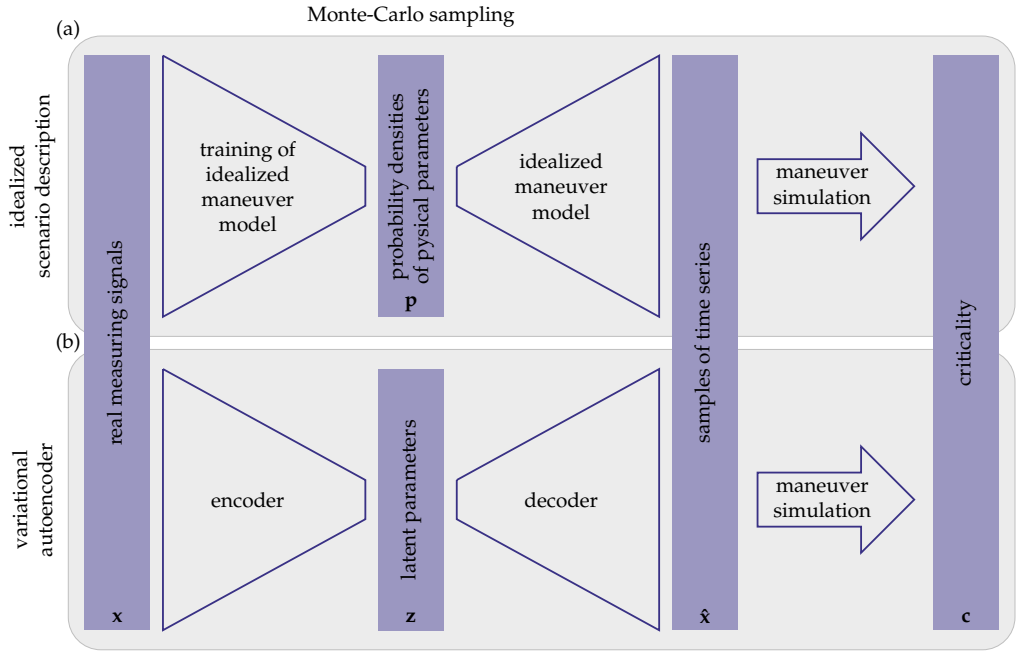

Figure 3. Generation of cut-in maneuvers by (**a**) idealized causal model and (**b**) AI model based on Monte Carlo simulation to determine the criticality of driver assistance systems.

In summary, the following problems and sources of error are associated with this idealized method:

- Low fitting errors require highly complex mathematical models describing maneuvers.
- Increasing mathematical complexity increases amount of input parameters to be identified.
- Modeling effort increases with number of logical scenarios to be investigated.
- Method is only applicable to clearly definable and separable driving situations.

Therefore, the present paper focuses on an alternative, AI-based approach for simulative validation of ADASs. In the newly developed methodology shown in Figure 3b, mathematical maneuver modeling is replaced by a generative AI model in the form of a variational autoencoder. This model independently learns an efficient representation of the measured data, where the encoder network maps the given real measuring signals $\mathbf{x}$ into a low-dimensional space of latent parameters $\mathbf{z}$, and the decoder network generates new but statistically identical samples of time series $\hat{\mathbf{x}}$. For safety assessment, Monte Carlo sampling may generate samples of latent variables $\mathbf{z}$ to be transformed to time series $\hat{\mathbf{x}}$

by the decoder and used for simulation analogously, as in the state-of-the-art approach. However, the newly developed AI model is expected to learn all the major features of real driving data and represent them through latent distributions and the decoder network better than the currently used mathematical model with physical parameters.

In order to apply this approach, the measured data need to be preprocessed, which is shown in Section 2 for the cut-in scenario. Next, the used variational autoencoder for generating realistic cut-in maneuvers is described in Section 3. For the validation of statistical properties, the proposed model is applied to the cut-in scenario in Section 4. Finally, the approximations by the AI model are compared with the current approach.

2. Data Preprocessing

The quality of a machine learning model depends largely on the data basis used. Therefore, suitable measurement signals for the considered driving maneuver must be selected and first preprocessed. The data are based on ECU signals from the ego vehicle and sensor information on objects in its immediate field of view. The control unit detects and records position and kinematics of such objects, where the available measurement data include, among others, the position (x_M, y_M) of an object relative to the ego vehicle and its absolute speed. The ECU has a sampling rate of 50 Hz, i.e., the time interval between two consecutive data points is 0.02 s.

In the context of this work, the position of an object is represented by so-called L-shapes derived from three corners framing the vehicle; see Figure 4a. From this simplified representation, the lateral position (x_M, y_M) of the center point relative to the center of the ego vehicle can be determined over the entire acquisition time. These raw signals (Figure 4b) need to be preprocessed in several ways:

(i) Correction of curved roads.
(ii) Smoothing and removal of dropouts.
(iii) Downsampling data points.

The position of an object relative to the ego vehicle is influenced by the course of the road. When a vehicle in front of the ego vehicle drives on a curve with radius R, the measured lateral distance y_M does not equal the lateral lane distance, which an ADAS has to react on; see Figure 4a.

In order to eliminate the curvature effect, the following geometric relations between measured relative coordinates (x_M, y_M) and the real longitudinal and radial distances (s, d) may be considered:

$$x_M = (R + d)\sin(\varphi), \quad y_M + R = (R + d)\cos(\varphi). \tag{1}$$

From the ratio of the two equations, we obtain the angle

$$\varphi = \arctan\frac{x_M}{R + y_M}, \tag{2}$$

and, e.g., from the second equation in Equation (1), the correction formula for the lateral lane distance

$$d = \frac{y_M + R}{\cos(\varphi)} - R. \tag{3}$$

The required curve radius R may be estimated from the derivate of $s = R\varphi$ resulting in

$$R = \frac{s}{\varphi} \equiv \frac{\dot{s}}{\dot{\varphi}} = \frac{v}{\omega}, \tag{4}$$

where actual ego velocity v and angular velocity ω are typically known quantities. It should be noted that Equation (3) is valid for both right curves ($R > 0$) and left curves ($R < 0$).

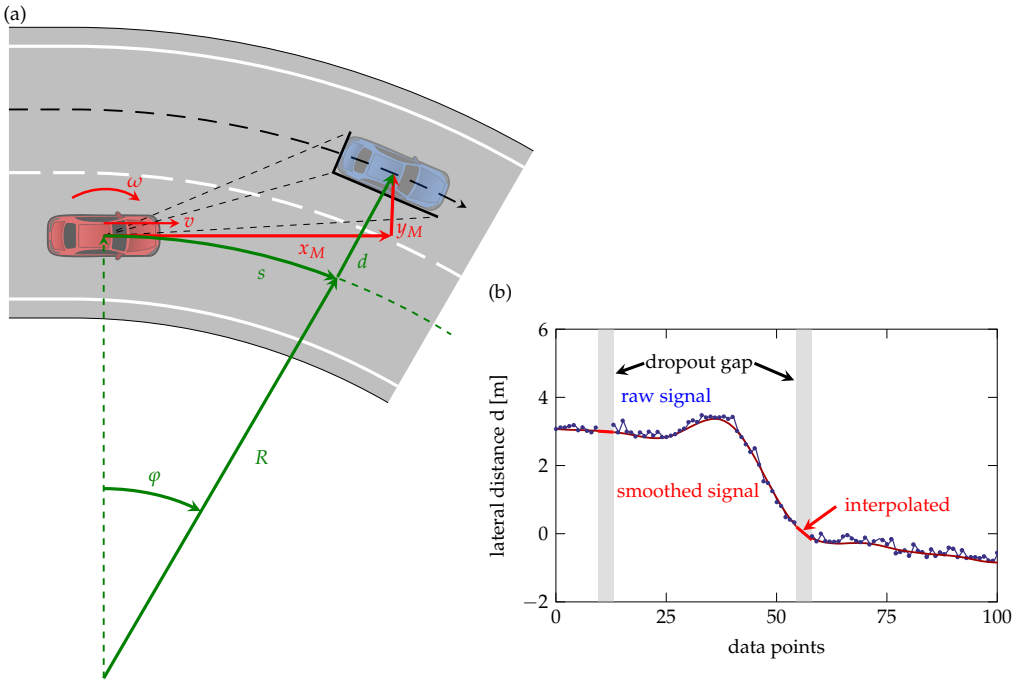

Figure 4. Data preprocessing: (**a**) correction of curved trajectory and (**b**) smoothing and interpolation of measured signal.

The recording of data in real driving tests is carried out in continuous series of measurements. However, due to the dynamics of a traffic situation and a constantly changing vehicle environment, gaps, jumps, and the superimposition of high-frequency noise may appear, as shown in Figure 4. Signals with such perturbations are not suitable, as training data for a generative AI model and must be removed [8,9]. In the following, signal snippets with a maximum length of $T = 20$ s are considered to ensure that a maneuver is complete. For signal gaps shorter than a tolerated length $\Delta t = \tau T$ (e.g., $\tau = 0.02$), linear interpolation between margin values is performed to reconstruct the missing signal values. While this approach can be assumed to be sufficient in the case of short signal gaps due to vehicle inertia, signals with larger gaps are discarded. In Figure 4b, two gaps are highlighted in gray and the corresponding interpolation in red.

In addition to missing measured values, implausible signal changes in particular cause a reduction in signal quality. Therefore, smoothing with a Savitzky–Golay filter is used, which performs a local polynomial regression. Here fourth-order polynomials using a window with a fixed width of 13 signal values are sliding through the signal [10]. After interpolation and application of the Savitzky–Golay filter, the red curve in Figure 4b has no discernible noise anymore.

For the majority of common machine learning algorithms, it is necessary that all input data have the same dimension, which should not be too high, either. Filling missing signal values with predefined values such as zeros (zero padding) would correspond to incorrect maneuver characteristics. Therefore, and in order to reduce the number of data points, signals $x_j(t),\ t \in [0, T_j]$, with measured length T_j, are downsampled to a fixed size $N_t = 100$ data points by resampling the time series to $x_{ij} = x_j(t_i)$ at time points

$$t_i = \frac{T_j}{N_t} \cdot i, \quad i = 0, ..., N_t, \tag{5}$$

based on linear interpolation between measured data.

3. Variational Autoencoder for Generating Realistic Cut-In Maneuvers

The dataset presented in the previous section is used for training a generative AI model with the structure shown in Figure 5. Formally, this task can be described as follows: The training data $\boldsymbol{x}$ result from an unknown distribution $p(\boldsymbol{x})$ which must be learned by the generative model in order to be able to generate new comparable data $\tilde{\boldsymbol{x}}$ that follow the same probabilistics $p_\theta(\tilde{\boldsymbol{x}}) \approx p(\boldsymbol{x})$. To generate new samples, a random sample $\boldsymbol{z} \in \mathbb{R}^d$ is taken from a known distribution $p_z(\boldsymbol{z})$ and then transformed by the generator $D(\boldsymbol{z};\, \boldsymbol{\theta})$ with internal parameters $\boldsymbol{\theta}$ to be learned. In principle, any generative model such as normalizing flows or generative adversarial networks may be used for this generative process. However, the different models are differently suited for specific tasks. Here, a variational autoencoder is used for the described process, which is briefly explained below.

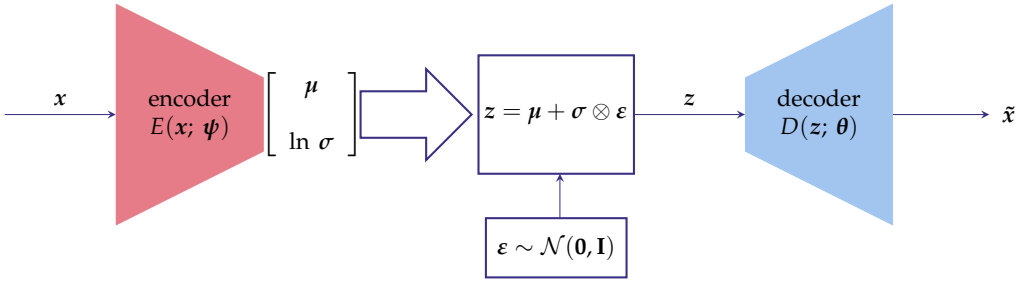

Figure 5. Structure of used VAE with dimension d of latent space.

A variational autoencoder (VAE) is a type of generative model that can learn to generate new data by capturing the underlying distribution of the training data [11,12]. VAEs are a variant of autoencoders consisting of an encoder and a decoder, as shown in Figure 5. The main idea behind VAEs is to learn a low-dimensional latent representation of the input data that captures the essential features and variations in the data. This latent representation can then be used to generate new samples that resemble the original data distribution [13].

More precisely, the encoder part $E(\boldsymbol{x};\, \boldsymbol{\psi})$ of a VAE maps the input data $\boldsymbol{x}$ to a latent space representation $\boldsymbol{z}$. It typically consists of several layers of a neural network that progressively reduce the dimensionality of the data, ultimately producing the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of a multivariate Gaussian distribution in the latent space. The encoder can be represented as a function of $\boldsymbol{x}$, generating the mean and the natural logarithm of the variance, i.e.,

$$E(\boldsymbol{x};\boldsymbol{\psi}) = \begin{bmatrix} \boldsymbol{\mu} \\ \ln\, \boldsymbol{\sigma} \end{bmatrix} \tag{6}$$

where parameters $\boldsymbol{\psi}$ summarize weights and biases of the artificial neural network. To generate a sample in the latent space, we need to obtain a latent vector $\boldsymbol{z}$ that follows the desired distribution. However, to allow training by backpropagation and stochastic gradient descent, we cannot directly sample from the z-distribution. Instead, the reparametrization trick addresses this challenge by introducing a separate normally distributed random variable $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is drawn from a multivariate standard Gaussian distribution. The vector $\boldsymbol{z}$ in the latent space is then obtained by

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \otimes \boldsymbol{\varepsilon}, \tag{7}$$

where $\otimes$ is an element-wise multiplication of the two vectors $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$.

The decoder part $D(\boldsymbol{z};\, \boldsymbol{\theta})$ of the VAE takes the latent vector $\boldsymbol{z}$ and maps it back to the input space, aiming at a reconstruction of the original data. It also consists of several layers of an artificial neural network with weights and biases summarized in parameter vector $\boldsymbol{\theta}$ to upsample the latent vector and eventually generate a reconstructed output $\tilde{\boldsymbol{x}}$.

Now, with both parts of the neural network architecture, an end-to-end training of the VAE by a joint optimization, also called evidence lower bound optimization (ELBO), can be performed [14]. Due to the random character of generating $\boldsymbol{z}$, outputs $\tilde{\boldsymbol{x}}$ and inputs $\boldsymbol{x}$ cannot be compared on a one-to-one basis but only statistically. Therefore, the loss function consists of two terms: the reconstruction loss $MSE = mean_k || \tilde{\boldsymbol{x}}^{(k)} - \boldsymbol{x}^{(k)} ||^2$, which measures how well the decoder can reconstruct the input data, and a regularization term, typically the Kullback–Leibler divergence (D_{KL}), which encourages the approximate posterior distribution $q_{\boldsymbol{\psi}}(\boldsymbol{z} \mid \boldsymbol{x})$ to be close to a prior distribution $p_z(\boldsymbol{z})$:

$$\mathrm{L}(\boldsymbol{\psi}, \boldsymbol{\theta}) := MSE + D_{KL}(q_{\boldsymbol{\psi}}(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p_z(\boldsymbol{z})), \tag{8}$$

During the training process, the parameters of the encoder ($\boldsymbol{\psi}$) and decoder ($\boldsymbol{\theta}$) networks are learned by minimizing this loss function using gradient descent or a similar optimization algorithms like ADAM. Once the VAE is trained, it can generate new samples by sampling from the prior distribution $p_z(\boldsymbol{z})$ and passing them through the decoder only.

Since the existing training data are multivariate time series signals, the model used must be able to correctly capture temporal dependencies within the data and take them into account when generating new data. Therefore, neural network topologies are needed that can recognize and learn temporal patterns in the training data. Both, convolutional neural networks and recurrent neural networks may be considered for the design of the variational autoencoder, since both network structures have proven to be suitable for tasks related to time series signals [15,16]. In the following, convolutional neural networks are chosen, as shown in Figure 6.

The training data comprise $N_f = 3$ different features consisting of time series

$$\boldsymbol{x} = \begin{bmatrix} \{t_i\} \\ \{d_i\} \\ \{v_i\} \end{bmatrix}, \quad i = 0, ..., N_t, \tag{9}$$

namely time points t_i, lateral lane distances d_i, and velocities v_i of the cut-in vehicle. The training dataset consists of samples $\boldsymbol{x}^k$ with $k \in 1, 2, ..., N_s$, shown in Figure 2. At this point, it should be mentioned that in the context of this work, the focus is on the lateral positions d_i.

In the first layer of the encoder, each of the three input channels is weighted with a sliding 3×1 convolution filter again producing time series with 100 elements, which are then summed up and biased to produce a new channel; and 100 such operations are performed to finally end up with 100 intermediate results, which are then processed element-wise with the nonlinear Rectified Linear Unit (ReLU) $\sigma_{\mathrm{ReLU}}(x) = max\{0, x\}$ as the activation function. The second and third layer operate in a way with reduced numbers of filters, reducing the number of channels first to 50 and then to 25. The latter output is flattened to a 25× 100 = 2500-dimensional vector and connected to the encoder output $\left[\boldsymbol{\mu}^T, \ln\ \boldsymbol{\sigma}^T\right]^T \in \mathbb{R}^{2d}$ by a fully connected network, where d is the dimension of the latent space.

As shown in Figure 6, the decoder has an almost symmetric structure in reverse order. Instead of the convolutional layers, transposed convolutional layers are used. The output of the decoder network corresponds to the reconstruction of the sample that was used as input for the variational autoencoder. The random input $\boldsymbol{z} \in \mathbb{R}^d$ is first enlarged to a 2500-dimensional vector by a fully connected layer, which is then reshaped to 25 channels of time series with 100 data points, respectively. This is enlarged to 50 and 100 channels by transposed convolution with 3 × 1 filters and ReLU activation functions, respectively. Finally, the 100 channels are reduced to three channels by convolution with 3 × 1 filters.

The total number of network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ of the variational autoencoder sums up to 227,603. Their training is carried out by an ADAM optimizer for a total of 700 epochs, where each epoch splits the total set of N_s input examples into minibatches of size 32. In order to improve training performance [17], the features in Equation (9) living on different

scales are initially normalized with respect to signal amplitudes, respectively, such that all features have the same range $[-1;1]$:

$$x_{ij}^{(k)} := -1 + \frac{2(x_{ij}^{(k)} - x_j^{min})}{x_j^{max} - x_j^{min}}, \qquad x_j^{min} = min_{ik}\, x_{ij}^{(k)}, \quad x_j^{max} = max_{ik}\, x_{ij}^{(k)},$$
$$i = 0, ..., N_t, \quad j = 1, ..., N_f, \quad k = 1, ..., N_s. \tag{10}$$

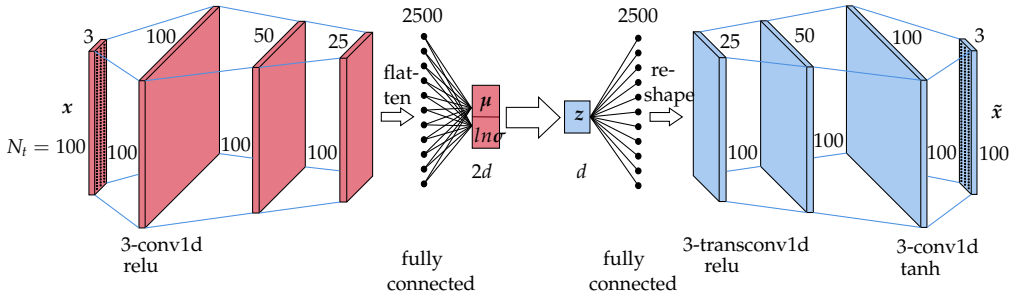

Figure 6. Structure of used VAE with dimension d of latent space.

For comparibility of the output $\tilde{\boldsymbol{x}}$ with this normalized input $\boldsymbol{x}$, the $tanh(x) \in (-1, 1)$ is applied element-wise to the last layer as the activation function. In order to generate trajectories similar to the measured ones in Equation (9), these outputs $\tilde{\boldsymbol{x}}$ need to be a rescaled as follows:

$$\tilde{x}_{ij}^{(k)} := \frac{x_j^{min} + x_j^{max}}{2} + \frac{x_j^{max} - x_j^{min}}{2} \tilde{x}_{ij}^{(k)}, \quad i = 0, ..., N_t, \quad j = 1, ..., N_f, \quad k = 1, ..., N_s. \tag{11}$$

4. Statistical Validation of VAE

The result of applying the above concept to cut-in maneuvers with latent space dimension $d = 10$ is not only optimal encoders (and especially decoders) for generating realistic maneuvers but also a sample $\boldsymbol{\mu}^{(k)}$, $\ln \boldsymbol{\sigma}^{(k)}$, $k = 1, ..., N_s$, with stochastic properties representing those of the input shapes $\boldsymbol{x}_{ij}^{(k)}$, $k = 1, ..., N_s$. Kernel Density Estimation (KDE) [17,18] may be used to fit coordinate-wise probability densities $p_\mu(\mu_m)$ and $p_\sigma(\ln\ \sigma_m)$, $m = 1...d$ to the elements of the encoder output in Figure 6. Some examples of these probability densities are shown in Figure 7.

These density functions highlight that the encoder outputs for mean (Figure 7a) and logarithmic variance (Figure 7b) do not always follow a standard Gaussian distribution. Only μ_1 and μ_8, approximately follow a Gaussian distribution, whereas, e.g., $\ln\ \sigma_{10}$, looks like a mixture of two Gaussian distributions.

The process of generating statistically correct approximation samples $\{\tilde{\boldsymbol{x}}_i^{(k)}, \ k = 1, ..., N_s\}$, which is important for correct failure assessment of ADASs, is then as follows:

(i) Chose $\boldsymbol{\mu}^{(k)}, \boldsymbol{\sigma}^{(k)}$ according to their densities like those in Figure 7;
(ii) Chose $\boldsymbol{\varepsilon} \sim \mathbb{N}(\boldsymbol{0}, I)$ according to a standard Gaussian distribution;
(iii) Superpose these quantities according to Equation (7) to obtain a sample $\{\boldsymbol{z}^{(k)}, \ k = 1, ..., N_s\}$ of latent variables;
(iv) Transform these latent variables with the decoder into normalized trajectories $\{\tilde{\boldsymbol{x}}^{(k)} = D(\boldsymbol{z}^{(k)}; \boldsymbol{\theta}), \ k = 1, ..., N_s\}$;
(v) Rescale these trajectories by Equation (11).

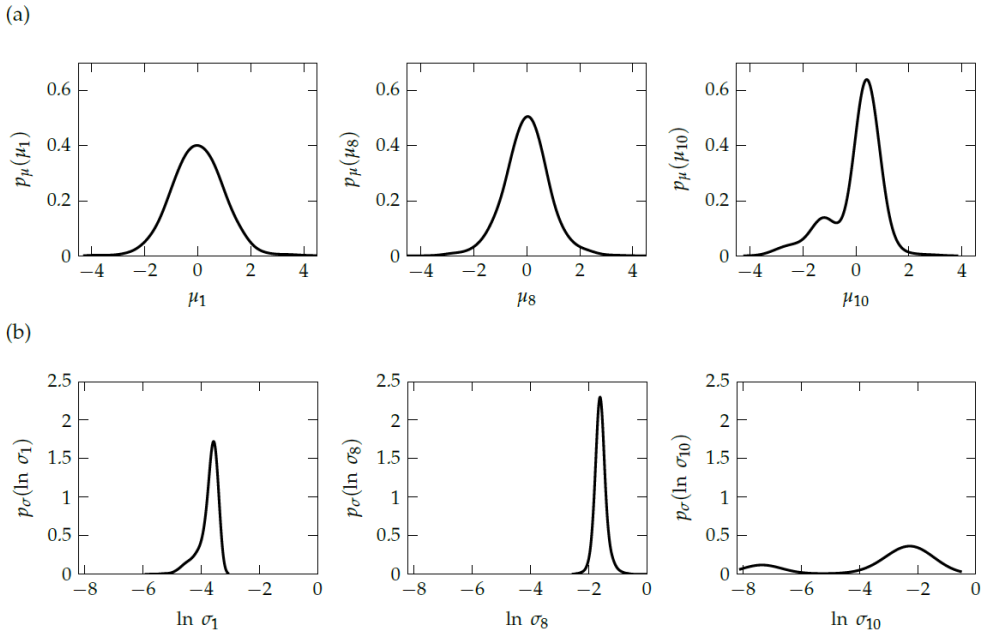

Figure 7. Exemplary probability density functions of (**a**) mean and (**b**) variance for coordinates $m \in \{1, 8, 10\}$ of latent space.

Figure 8a shows the set of the same number as Figure 2 of randomly generated trajectories.

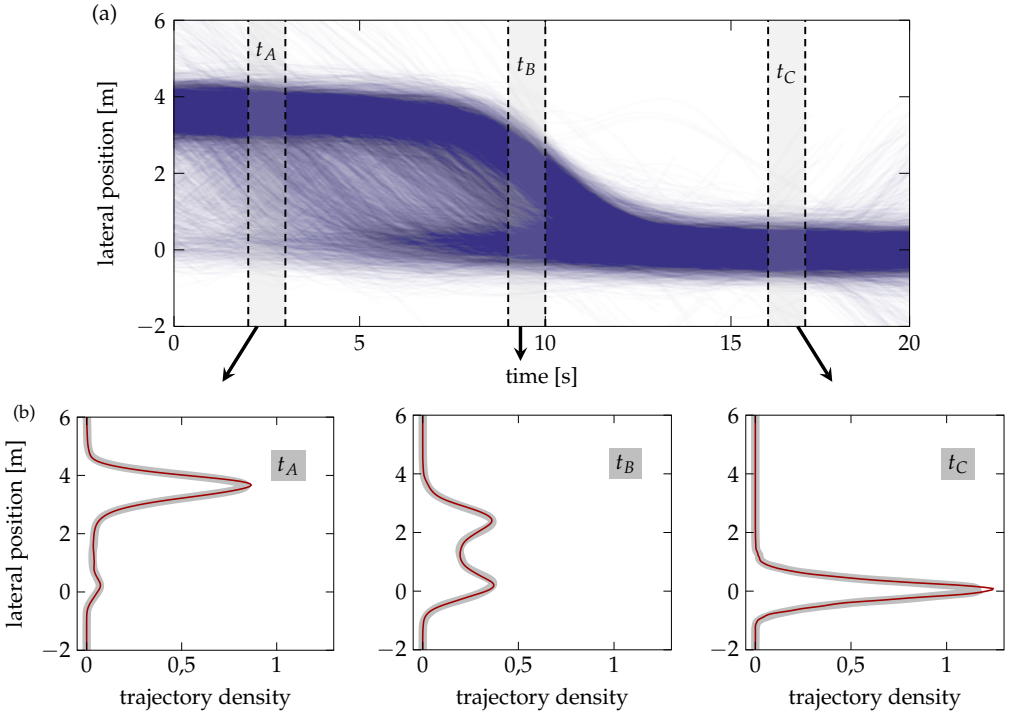

Figure 8. Generated lateral trajectories by decoder of the trained VAE (**a**) and corresponding densities (**b**) of real (underlying gray line) and generated (red line) trajectories for a time stripe of 1s width at various time instances.

Obviously, the high-intensity areas of the lateral position over time match very well with the real driving data. From this visual inspection, it can already be concluded that the probability of occurrence for specific driving situations is learned by the model and maintained when generating new data samples. In order to check quantitatively whether

the synthetic data have equal statistical properties, density functions for lateral position are determined with a KDE for three different time strips of 1s width. Figure 8b proves that the statistical properties are almost equal for the time periods t_A, t_B, and t_C. This property is essential for the validation of ADASs on the basis of a randomized generation of test cases. If, for example, the model would give preference to more critical driving situations, which would then account for a much larger proportion of all test cases generated than is the case in the real driving data, the probability of failure of ADASs would be overestimated.

For a check of the statistical properties with respect to time behavior, another analysis is applied. For every data point $\boldsymbol{d}_i$ of the lateral distance time series, the mean and variances

$$\boldsymbol{\mu}_{d,i} = mean_k\ \boldsymbol{d}_i^{(k)}, \quad \boldsymbol{\sigma}_{d,i} = var_k\ \boldsymbol{d}_i^{(k)}, \quad i = 1, ..., N_t, \tag{12}$$

are estimated. The resulting curves for mean and variance of the lateral position d are illustrated in Figure 9.

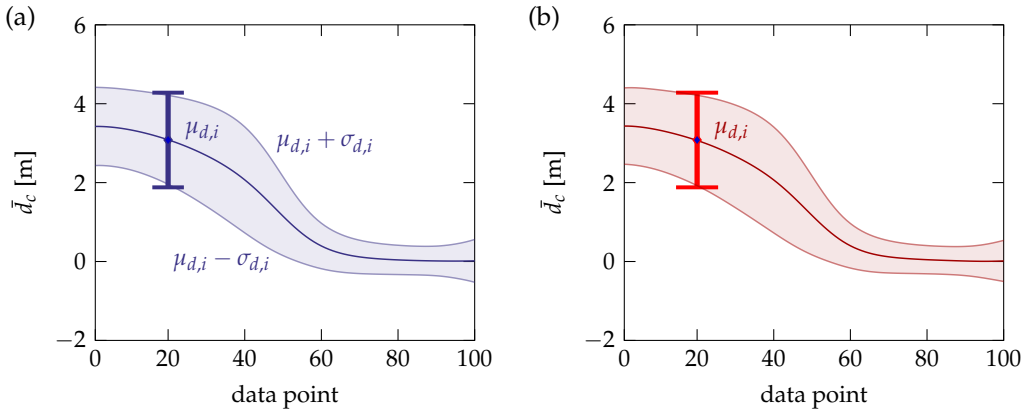

Figure 9. Mean and variation of (**a**) measured and (**b**) generated trajectories.

The filled areas mark the intervals $[\mu_{d,i} - \sigma_{d,i},\ \mu_{d,i} + \sigma_{d,i}]$, $i = 1, ..., N_t$, for the measured cut-in data (Figure 9a) and the generated trajectories (Figure 9b). At the beginning of the maneuvers, the variation is higher than at the end, since the start point of the cut-in maneuver can differ within the lane widths on highways, whereas the goal of each maneuver is to reach the center of the ego lane at the end. Nevertheless, the comparison of measured and AI-generated trajectories highlights that the regions along time are almost identical, and no differences are visible.

Next, we may investigate if the generated trajectories not only fit statistical properties but also reflect the time characteristics of, e.g., the real cut-in maneuver highlighted in Figure 2. Unfortunately, a direct comparison of a generated trajectory with a given real one is not possible due to the probabilistic character of VAEs. Therefore, the nearest neighbor of the real trajectory within the generated dataset $\{\tilde{\boldsymbol{x}}^{(k)}\}$ is searched, where the Euclidean distances of all data points of the time series are used as distance measure. Figure 10 shows the original trajectory from Figure 2 (blue dashed) and the closed neighbor (red) of all the synthetic motion curves in the generated dataset, which has great similarity to the measured cut-in trajectory.

It is particularly important to emphasize that this is a randomly generated curve and not just a reconstruction of the real trajectory determined, e.g., with a deterministic autoencoder. From the comparison shown, it can be concluded that the generative model is able to depict the real lateral trajectory more realistically than, e.g., the mathematical model [6], using a third-order polynomial (black) and about 20 parameters, even though the latter curve was found by optimization as the closest approximation of the given measured trajectory. While the mathematical model can only depict the basic course of the lane change, the generative model can reproduce all the specific characteristics of the maneuvers. With about the same number of parameters being $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, the AI model can

apparently depict the real driving data more realistically than the mathematical model currently used.

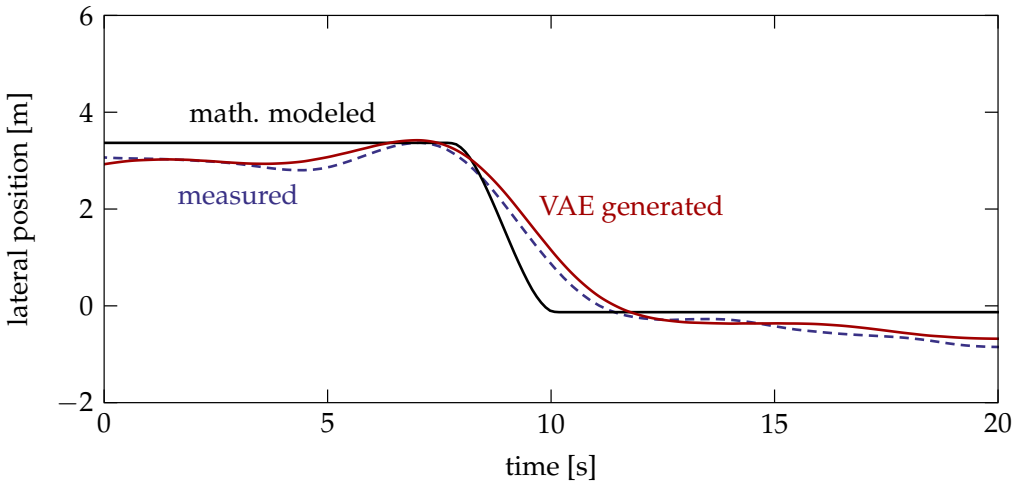

Figure 10. Comparison between measured (**blue dashed**), mathematically modeled (**black**), and AI-generated (**red**) lateral trajectory.

5. Conclusions

Simulative validation plays a major role in the validation of highly automated driver assistance systems, such as the Drive Pilot from Mercedes-Benz AG, which was the first Level-3 system (according to SAE standard) to be approved in Germany. This requires the generation of safety-relevant driving situations with the same characteristic and stochastic properties as real traffic scenarios. The currently applied idealized models are not able to fulfill these needs, whereas the AI-based concept presented in this paper can generate highly variable maneuvers of the same motion type and with the same probabilities as those observed in reality. Statistical validation has shown that both the time characteristics and the statistical properties fit those of the input data. It only requires the training of a variational autoencoder with measured data and no physical understanding or any high-level mathematical modeling effort. Therefore, the concept may be applied to any other common driving scenario, such as cut-out or cut-through scenarios, and used as a basic tool to support the further development of highly automated driving functions.

6. Patents

A patent application is in preliminary examination.

Author Contributions: Conceptualization, Z.K. and D.B.; Methodology, Z.K.; Software, Z.K.; Investigation, Z.K. and R.R.; Writing – original draft, Z.K. and D.B.; Project administration, P.B. All authors have read and agreed to the published version of the manuscript.

Funding: The second author is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Project No. 501840485.

Data Availability Statement: Not applicable.

Conflicts of Interest: Authors Zafer Kayatas, Pascal Bestle and Robin Reick were employed by the company Mercedes-Benz AG, Sindelfingen, Germany. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Winner, H.; Hakuli, S.; Lotz, F.; Singer, C. *Handbuch Fahrerassistenzsysteme*; Springer: Wiesbaden, Germany, 2015.
2. PEGASUS. Anforderungen und Rahmenbedingungen—Stand 4. Szenarienbeschreibung. Available online: https://www.pegasusprojekt.de/en/pegasus-method (accessed on 21 June 2022).

3. Roesener, C.; Fahrenkrog, F.; Uhlig, A.; Eckstein, L. A Scenario-Based Assessment Approach for Automated Driving by Using Time Series Classification of Human-Driving Behaviour. In Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016.
4. Roesener, C.; Harth, M.; Weber, H.; Josten, J.; Eckstein, L. Modelling Human Driver Performance for Safety Assessment of Road Vehicle Automation. In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018.
5. Pfeffer, R. Szenariobasierte Simulationsgestützte Funktionale Absicherung Hochautomatisierter Fahrfunktionen Durch Nutzung von Realdaten. Ph.D. Thesis, Karlsruher Institut für Technologie, Karlsruhe, Germany, 2020.
6. Kayatas, Z. Optimierung der Zuverlässigkeitsanalyse für die Simulative Absicherung Hochautomatisierter Fahrerassistenzsysteme. Master's Thesis, Leibniz Universität Hannover, Hannover, Germany, 2020.
7. Rasch, M.; Ubben, P.T.; Most, T.; Bayer, V.; Niemeier, R. Safety Assessment and Uncertainty Quantification of Automated Driver Assistance Systems using Stochastic Analysis Methods. In Proceedings of the NAFEMS World Congress, Tampa, FL, USA, 15–18 May 2019.
8. Pirouz, A. Identifizierung Kritischer Fahrszenarien aus Messdaten mittels Machine Learning Methoden. Master's Thesis, Stuttgart Universität, Stuttgart, Germany, 2021.
9. Reichenbächer, C.; Rasch, M.; Kayatas, Z.; Wirthmüller, F.; Hipp, J.; Dang, T.; Bringmann, O. Identifying Scenarios in Field Data to Enable Validation of Highly Automated Driving Systems; *arXiv* **2022**, arXiv:2203.03515.
10. Gallagher, N. Savitzky-Golay Smoothing and Differentiation Filter. Available online: https://eigenvector.com/wp-content/uploads/2020/01/SavitzkyGolay.pdf (accessed on 11 December 2022).
11. Tomczak, J. *Deep Generative Modeling*; Springer Nature Switzerland AG: Cham, Switzerland, 2022.
12. Hinton, E.; Salakhutdinov, R. *Reducing the Dimensionality of Data with Neural Networks*; Technical Report; 2006. Available online: https://www.cs.toronto.edu/~hinton/absps/science.pdf (accessed on 15 August 2023).
13. Kingma, D.; Welling, M. *An Introduction to Variational Autoencoders*; Technical Report. 2019. Available online: https://arxiv.org/pdf/1906.02691.pdf (accessed on 15 August 2023).
14. Kingma, D.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; Welling, M. *Improved Variational Inference with Inverse Autoregressive Flow*; Technical Report; 2016. Available online: https://arxiv.org/pdf/1606.04934.pdf (accessed on 15 August 2023).
15. Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P. *Deep Learning for Time Series Classification: A Review*; Data Mining and Knowledge Discovery 33; 2019. Available online: https://link.springer.com/article/10.1007/s10618-019-00619-1 (accessed on 15 August 2023).
16. Langner, J.; Bach, J.; Ries, L.; Otten, S.; Holzapfel, M.; Sax, E. *Estimating the Uniqueness of Test Scenarios Derived from Recorded Real-World-Driving-Data Using Autoencoders*; Technical Report. 2018. Available online: https://ieeexplore.ieee.org/document/8500464/authors#authors (accessed on 15 August 2023).
17. Parzen, E. On estimation of a probability density function and mode. *Ann. Math. Stat.* **1962**, *33*, 1065–1076. [CrossRef]
18. Chen, Y.C. A Tutorial on Kernel Density Estimation and Recent Advances. *arXiv* **2017**, arXiv:1704.03924.