2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Data-Driven Maneuver Modeling using Generative Adversarial Networks and Variational Autoencoders for Safety Validation of Highly Automated Vehicles

Robert Krajewski[1], Tobias Moers[2], Dominik Nerger[2] and Lutz Eckstein[1]

***Abstract*— Scenario-based validation is a promising approach for the safety validation of highly automated driving systems. By modeling relevant driving scenarios, utilizing simulations and selecting insightful test cases, the testing effort is reduced. However, current methods can't automatically create intuitive models of the vehicle trajectories in a scenario. We propose to use unsupervised machine learning to train neural networks to solve the modeling problem. The models learn a small set of intuitive parameters without the need for labeled data and use them to generate new realistic trajectories. The neural networks, which base on the InfoGAN and beta-VAE architectures, are adapted from the image domain to the time series domain. Although our methods are generally applicable, our experiments focus on lane change maneuvers on highways. To train the networks, we use more than 5600 measured lane change trajectories extracted from the highD dataset. Our results show that the networks learn to describe lane change maneuvers by up to four intuitive parameters. Furthermore, the networks are able to map existing lane change trajectories to values of the learned parameters and generate new, previously unseen, realistic trajectories. We compare both architectures among themselves and to a polynomial model, and show respective advantages.**

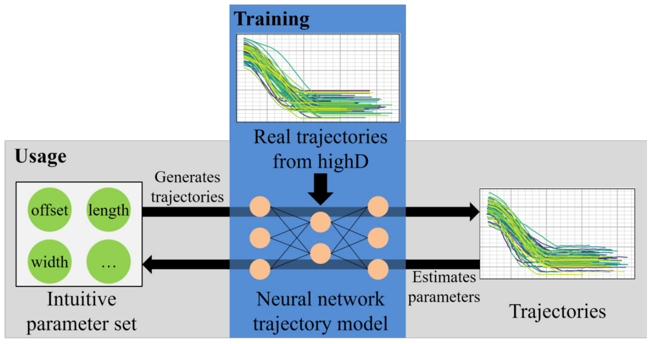


Fig. 1: Our proposed methods use real trajectories from the highD dataset to train neural networks that generate synthetic trajectories from intuitive parameters as input. At the same time, the neural networks can perform the inverse mapping from given trajectories to corresponding parameter values.

I. Introduction

The increasing amount of highly automated vehicle (HAV) prototypes and research in related areas show that many technical challenges of automated driving are tackled or can already be solved adequately. While the performance of algorithms and systems is steadily increasing, the safety validation of HAVs remains an open issue. Existing methods, such as extensive test drives on public roads and reproducible tests on test tracks, are not feasible [1]. The validation effort typically increases with the amount and complexity of the scenarios a system has to handle. According to [2], more than 6.6 billion kilometers of test driving would be necessary to ensure the safety of a HAV. This makes the aforementioned ordinary methods unfeasible because of cost and time efforts. Public research projects such as PEGASUS [3] or ENABLE-S3 [4] try to tackle the safety validation challenge by increasing testing in simulation environments and introducing a scenario-based approach. The key idea of this approach is to define all relevant driving scenarios a HAV can encounter within the defined system boundaries. Here, a scenario completely defines the infrastructure and the vehicles' driven maneuvers except for the vehicle that is tested [5]. For these scenarios, among other things, the occurrence probability and the performance of a human driver are derived from data like driving studies or accident studies. This information allows to reduce the testing effort. For example, test cases introducing no new insights about the safety of the system could be skipped.

The issue of defining a valid set of driving scenarios and scenario elements for the safety validation of a HAV has not been solved yet. Previous research projects have derived scenarios subsets that include only specific scenarios such as vehicle collisions [6]. These are not sufficient for testing HAVs, as other scenarios (e.g. non-collision scenarios) have to be tested as well. In [7], a systematic approach for generating scenarios is presented, which is based on combining scenario elements. These elements are categorized into five different layers and include infrastructure elements, vehicles or weather conditions. For static infrastructure elements, the set of relevant elements and possible variations can be derived from multiple sources such as map data or national guidelines. However, the set of possible maneuvers which describe a vehicle's movements is typically created manually by experts. This applies for the modeling of the maneuver trajectory and choosing the parameters, e.g. the duration of a lane change, as well.

[1]The authors are with the Automated Driving Department, Institute for Automotive Engineering RWTH Aachen University (Aachen, Germany). (E-mail: {krajewski, eckstein}@ika.rwth-aachen.de).
[2]The authors are with the Automated Driving Department, Forschungsgesellschaft Kraftfahrwesen mbH Aachen (Aachen, Germany). (E-mail: {tobias.moers, dominik.nerger}@fka.de).

For example, modeling lane changes requires to manually analyze measured trajectories, choose a mathematical function to describe the shape of the trajectory, and find intuitive parameters [8]. These steps require effort and often a trade-off between accuracy and intuitiveness exists.

We propose a new approach to create models of maneuver trajectories based on unsupervised machine learning algorithms. A neural network learns to model maneuver trajectories and to find a comprehensible set of model parameters from recorded trajectories (see Fig. 1). The resulting network can generate any number of new realistic trajectories by varying the values of the model's input parameters. Additionally, the inverse mapping from trajectories to parameter values is learned by the network. Thus, the parameter representation of a rare or critical maneuver can be derived and varied to generate an infinite number of relevant trajectories for testing. This is especially useful, as datasets typically contain mostly common variations of maneuvers but rarely include critical variations.

We implement and compare two approaches. The first approach is based on a neural network architecture called InfoGAN [9], which is an extension of the Generative Adversarial Network (GAN) [10]. GANs are a recent approach for unsupervised generative modeling and achieve state-of-the-art (SOTA) results in different domains of machine learning. Especially in the image domain, GANs are applied for tasks such as domain transfer of images [11], [12] or image manipulation [13]. The second approach is based on the β-VAE architecture [14], [15], which is an extension of the original Variational Autoencoder (VAE) [16]. Despite the lower quality of the generated images, VAEs are easier to train and β-VAE has proven its ability to create a more intuitive representation on images. We adapt these networks to handle time series data of vehicle trajectories consisting of x/y coordinates. To train our network, we need a dataset of realistic trajectories. For this paper, we choose the highD dataset [17], which is a recent dataset of vehicle trajectories on German highways extracted from drone recordings. We focus on lane change maneuvers, as more than 5600 of these are already annotated in the highD dataset. In general, models e.g. based on polynomials can be used to approximate lane changes sufficiently well [8], [17]. However, our method is not constrained to lane changes, but can be used on every possible maneuver.

The main contributions are as follows:

- We propose two neural network architectures, called Trajectory Generative Adversarial Network (TraGAN) and Trajectory Variational Autoencoder (TraVAE), that are able to generate a disentangled and meaningful representation of lane change maneuver trajectories from recorded lane changes without any labels.
- We demonstrate that the network architectures are indeed able to recognize existing and generate new maneuver trajectory variations that are realistic.
- We compare both architectures and show their respective (dis-)advantages.
- We show that the networks can be used to find gaps in existing training or validation data and fill those gaps by generating realistic trajectories for use in e.g. simulation environments.

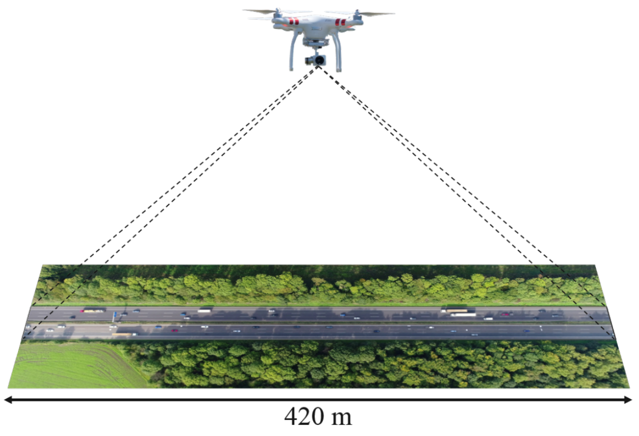


Fig. 2: Recording setup of the highD dataset [17].

II. Related work

A. The highD Dataset

The highD dataset [17] consists of vehicle trajectories on German highways. Using a consumer grade drone, highway traffic has been recorded from a bird's eye perspective (see Fig. 2) at six different sites around Cologne, Germany in 2017 and 2018. Computer vision algorithms have been used to detect and track the vehicles passing by to extract their trajectories. Typically, most automotive datasets are recorded using test vehicles equipped with state-of-the-art sensors [18], [19]. However, using a drone offers clear advantages. Firstly, the positions of the extracted vehicles and infrastructure are very accurate. While vehicle-equipped or infrastructure-equipped sensors suffer from problems like occlusion or decreasing accuracy with increasing distance, these problems do not exist when recording from a bird's eye perspective. Recording videos in 4K resolution allows to localize vehicles with high accuracy of about 10 cm (see Fig. 3). As a road segment of more than 400 m is covered, more than 20 vehicles in both directions are recorded on average at the same time. Thus, from every hour of recorded highway traffic, more than 1000 driven kilometers can be extracted. Finally, trajectories from drone videos are naturalistic. While test vehicles could influence surrounding vehicles with their conspicuous sensors, a drone is not noticeable to drivers.

The highD dataset includes 60 recordings with a total length of more than 16 hours on highways with two or three driving lanes per direction. The recordings contain about 110 000 vehicles split into about 90 000 cars and 20 000 trucks. This is about ten times as much data compared

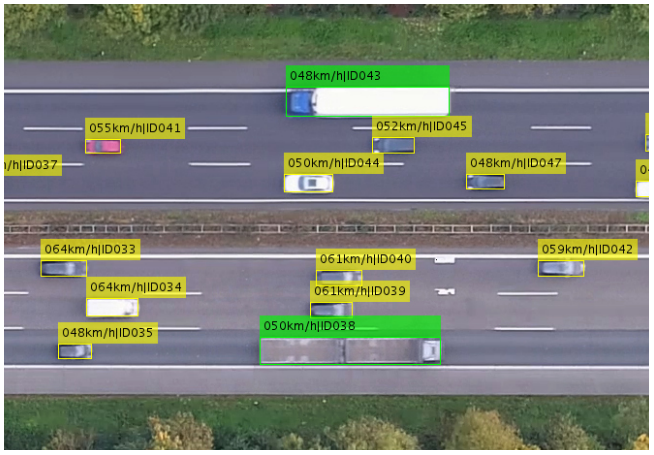


Fig. 3: Example of vehicle positions that have been extracted from recordings of the highD dataset [17].

to alternative datasets such as NGSIM [17], while also containing higher quality data. For each extracted vehicle, its preprocessed trajectory including positions, speeds and accelerations is given. In addition to that, the relations to surrounding vehicles, common metrics like Time-to-Collision, and basic driven maneuvers like lane changes are annotated by algorithms.

B. *Maneuver Trajectory Modeling*

Modeling maneuver trajectories is necessary for many purposes, including the generation of trajectories for the surrounding traffic in simulation environments or the ego vehicle motion planning. Depending on the use case, different requirements arise, e.g. a trajectory has to be comfortable while it is driven by the ego vehicle transporting passengers [8]. Maneuvers in simulation environments are commonly modeled using simple mathematical functions. Typically, the generated trajectory is used by a driver model or control algorithm as reference trajectory. In general, a trade off often exists between covering the variance of a maneuver and the resulting model's simplicity. For the safety validation of HAVs, is is necessary to model not only the common cases. But at the same time, the model must be intuitive to use in order to be utilized by an expert to generate specific trajectories.

However, it is a challenging task for an expert to create a model for each maneuver that is both comprehensive and intuitive. Thus, a fully automatic and unsupervised modeling process is desirable to model multiple maneuvers with ease. In this paper, we focus on lane change maneuver trajectories, as those are already well researched. They are typically modeled by (a combination of) simple mathematical shapes like circles, sinus functions and fifth degree polynomials [20]. Lane changes in the highD dataset are modeled by a quadratic polynomial for the longitudinal direction and a polynomial of fifth degree for the lateral direction, as more dynamics are present in the lateral direction [17]. In the longitudinal direction, most vehicles keep or constantly change their speed. As we assume that the generated trajectories are not used for the ego vehicle's but surrounding vehicles' motions, requirements such as comfort are neglected in the following.

C. *Generative Models*

Generative modeling describes the task to build a model that can generate new data that is similar to given training data. Typically, generative models allow to specify a set of parameters or statistics as input to control the data generation. In the following, we will also call these parameters latent parameters or latent variables. Assigning each latent parameter a concrete value yields a latent code, which is a complete latent representation of the data to be generated. The space of all possible latent codes is called latent space. The quality of a generative model can be assessed by the quality of the generated data samples and by the intuitiveness and disentanglement of the latent parameters. Disentanglement means that changing the value of a single parameter only changes a specific attribute of the generated samples. SOTA models in the generative modeling domain are based on neural networks and adapt their network architecture to the problem domain. In this section, we present the most common architectures.

1) Variational Autoencoder: A common approach for data generation is to use a Variational Autoencoder (VAE), which has been introduced by Kingma & Welling [16] as an extension of a regular autoencoder. An encoder network is trained to encode inputs to a low dimensional representation. To learn a meaningful latent representation, a decoder network has to reconstruct the original input using this representation. By training the network to minimize the difference between the network input and output, the latent space representation has to retain as much information as possible. After training, the decoder network is used to generate new data by sampling from the latent space. In contrast to a regular autoencoder, the VAE (see Fig. 4) additionally minimizes the volume occupied by the latent representations in the latent space. This is done by penalizing the distribution of latent codes to differ from a normal distribution using the Kullback-Leibler (KL) divergence. This avoids gaps in the latent space and improves the quality of generated samples.

However, a VAE typically does not generate an intuitive and disentangled latent space representation. Therefore, changing the value of a single parameter affects different attributes of the generated samples simultaneously. A solution to this problem is the β-VAE [14]. This extension to the standard VAE introduces a loss weight factor $\beta > 1$, which increases the importance of the KL divergence. Using a high value for β, the network is forced to encode the information of the dataset in a smaller subset of latent codes, which inherently leads to a disentangled representation.

2) Generative Adversarial Networks: Generative Adversarial Networks (GANs) are a new, but already commonly used unsupervised neural network architecture to create SOTA generative models in the image domain. GANs were

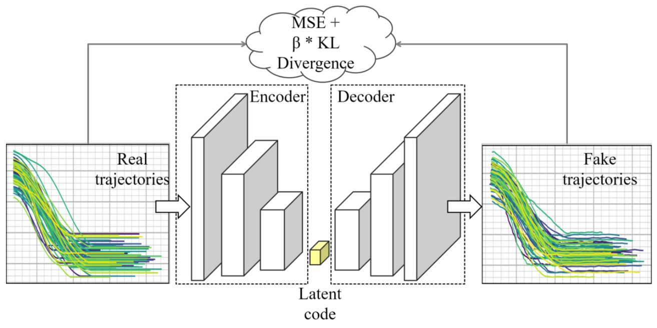


Fig. 4: The VAE is composed of two main components: The encoder and the decoder. The encoder creates a latent representation of the input and the decoder reconstructs the input from the latent representation. The loss of the network is defined as a combination of the reconstruction error and the Kullback-Leibler (KL) divergence.

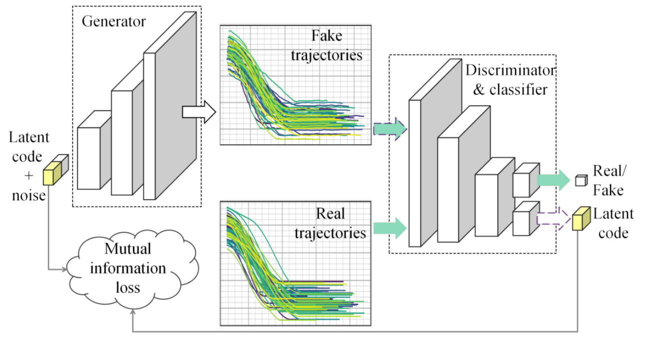


Fig. 5: The InfoGAN architecture consists of multiple networks. The generator network creates from a latent code and noise synthetic trajectories. The discriminator network learns to distinguish synthetic from given real trajectories. The classifier network shares most of its layers with the discriminator and estimates latent codes for given trajectories.

first introduced by Goodfellow in 2014 [10]. Comparable to autoencoders, GANs consist of a combination of two networks: A generator and a discriminator. The generator is the analogue to the decoder of an autoencoder and generates synthetic data samples from a latent representation. Its counterpart, the discriminator, has to distinguish between real and generated data samples. Both networks are trained alternately, which causes both networks to improve over time, until the generator is able to generate data samples that the discriminator is unable to distinguish from real data samples. Using a discriminator avoids the problem of blurry generated samples, which is a typical problem of autoencoders. The alternated training of generator and discriminator is not stable and sensitive to hyperparameters. Extensions like the WGAN-GP [21] help to improve the stability of the training.

Similar to a standard VAE, training a GAN typically does not result in disentangled latent parameters. The SOTA solution to this problem is the InfoGAN architecture [9] (see Fig. 5). An additional classifier network is used that estimates the latent code to generate given synthetic data samples. Maximizing the mutual information between these parameters and the data samples encourages the network to disentangle the input parameters.

III. Method

A. Data Selection and Preprocessing

As described in [17], the highD dataset not only provides the raw vehicle tracks, but also annotations for all lane changes. To create these, an algorithm detects vehicles crossing lane markings. All time samples before and after these crossings have been labeled as a lane change if the latitudinal movement is above a specified threshold. For every lane change, the trajectory of the according vehicle and meta information, like the duration or the direction of the lane change, are given. The trajectory itself consists of longitudinal and lateral positions, from which the speeds at every time step can be derived. We do not include double lane changes and incomplete lane changes. Additionally, we transform the coordinate system of the trajectories so that $x = 0$ is at the start of the lane change, $y = 0$ at the crossed lane marking and $y = -1/+1$ at the neighbouring lane markings. Finally, we mirror all lane changes that are facing to the left, so only lane changes to the right lane are used in the following.

In general, maneuver trajectories have a dynamic length. This also applies to the trajectories of lane changes (see Fig. 6), since the duration of the maneuver depends on factors such as the lateral speed of the lane changing vehicle or the lane width. The length of lane changes in the highD dataset ranges from 75 to 300 time steps at a sample frequency of 25 Hz. However, common neural network architectures can only handle inputs of a constant size. Specialized architectures like Seq2Seq can handle inputs of arbitrary size, but are harder to train [22]. Therefore, it is easier to harmonize the trajectory lengths by padding the end in this case. Instead of simply adding zeros, we extrapolate the longitudinal motion linearly while keeping the lateral position constant. As we

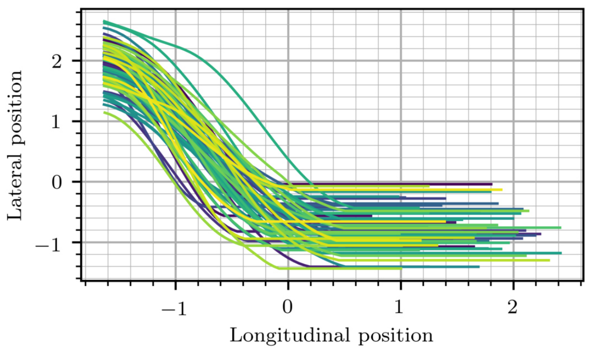


Fig. 6: Real lane change maneuver trajectories that have been sampled from the highD dataset.

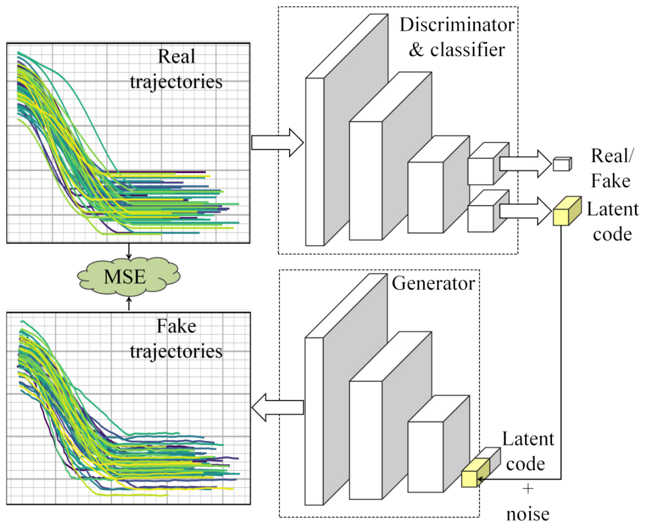


Fig. 7: The autoencoder, that has been built with the generator and classifier networks of our TraGAN, calculates the mean squared error between all real trajectories and the trajectories that have been restored from the estimated latent code.

are interested in the global motion and a unnecessarily long trajectory would make the training more difficult, the trajectories are sampled down by a factor of four.

B. Adaption of the InfoGAN

The standard InfoGAN is mostly used in the image domain and mainly consists of 2D convolution layers. As trajectories have a single temporal dimension instead of two spatial dimensions, we replace all 2D convolutions by 1D convolutions. In contrast to the standard InfoGAN, we exclusively use continuous latent parameters as input for the model. Instead of the standard GAN objective, we use the WGAN-GP objective [21] because it leads to a faster and more robust training.

To assess the quality of the generated trajectories and latent space representation, we repurpose the generator network and classifier network to function as an autoencoder comparable to [23]. As shown in Fig. 7, we feed real trajectories to the classifier network and combine the latent code output with a noise vector. The generator network creates a reconstruction of the input trajectory from the latent representation. The mean squared error between the input and reconstructed trajectory is a metric for the classification and generation quality of the network. Additionally, this loss is used for training the generator. It helps to create smooth trajectories, as we have noticed that the discriminator typically focuses on the global trajectory shape.

C. Adaption of the β-VAE

Similar to our adaption of the InfoGAN, we exclusively use continuous parameters for our adaption of the β-VAE. The design of TraVAE is almost identical to the original β-VAE [14]. But as trajectories are one-dimensional, 1D convolution layers replace 2D convolution layers. Instead of the cross-entropy loss, we use the mean squared error between original and reconstructed trajectories. Hence, we need to adapt the range of values for β as well [14].

To measure the trajectory quality for our TraVAE, we do not need to repurpose any part of the network, as the network already minimizes the mean squared error loss between the original trajectories and the reconstructed trajectories during training. Although, images reconstructed by autoencoders are typically blurry, this does not pose a problem for trajectories, as smooth trajectories are desired.

IV. Experiments

A. Experiment Setup TraGAN

The network layers for TraGAN are shown in Tables I and II. For the experiments shown in this paper, we use the Adam optimizer [24] with a learning rate of 1e-4 for the generator and 5e-4 for the discriminator and classifier. We use the WGAN-GP objective with a gradient penalty weight of 10 for all experiments. The weights for the mutual information loss and mean squared error loss are 2 and 30, respectively. Moreover, the TraGAN is trained with ten noise and eight latent parameters. While the latent parameters ideally represent distinctive attributes of the trajectories, the noise parameters are needed to allow minimal deviations.

B. Experiment Setup TraVAE

The network architecture of TraVAE is shown in Table IV. For the experiments that have been conducted with TraVAE, we use the Adam optimizer with a learning rate of 5e-4. We tested different sizes for the latent representation, ranging from 3 to 10 parameters. However, only a maximum of three latent parameters are intuitive and disentangled. The most crucial part of training the network is the choice of the right weight β for the KL divergence, which influences the degree of disentanglement between the latent parameters. If β is chosen too high, the KL divergence dominates the total loss and mode collapse occurs. Due to replacing the cross-entropy with a mean squared error loss, we use much lower values

TABLE I: Generator network architecture.

Layer	**Layer Information**
Input Layer	CONCAT-(NOISE, LATENT CODE)
Hidden Layer	FC-(N1024), ReLU
Hidden Layer	FC-(N(SL·64/10)), ReLU
Hidden Layer	RESHAPE-(BS, 1, $SL/10$, 64)
Hidden Layer	DECONV1D-(N64, K3, S2), ReLU
Hidden Layer	CONV1D-(N64, K3, S1), ReLU
Hidden Layer	DECONV1D-(N128, K5, S5), ReLU
Hidden Layer	CONV1D-(N128, K3, S1), ReLU
Hidden Layer	CONV1D-(N2, K3, S1)
Output Layer	RESHAPE-(BS, SL, FS))

Note - N: number of filters/neurons, SL: signal length, BS: batch size, , K: kernel size, S: stride, FS: feature size.

TABLE II: Discriminator and classifier network architecture.

Layer	Layer Information
Input Layer	RESHAPE-(BS, SL, 1, FS)
Hidden Layer	CONV1D-(N64, K7, S3), Leaky ReLU
Hidden Layer	CONV1D-(N64, K3, S1), Leaky ReLU
Hidden Layer	CONV1D-(N64, K3, S2), Leaky ReLU
Hidden Layer	CONV1D-(N64, K5, S1), Leaky ReLU
Hidden Layer	MAX POOL-(K10)
Hidden Layer	RESHAPE-(BS, -1)
Discriminator Layer	FC-(N1024), Leaky ReLU
Discriminator Output	FC-(N1), SIGMOID
Classifier Layer	FC-(N1024), Leaky ReLU
Classifier Layer	FC-(N64), Leaky ReLU
Classifier Output	FC-(N(LATENT DIM))

for β compared to the original implementation [14]. In our experiments, values in the range $[0.0001, 0.005]$ lead to the desired disentanglement of latent parameters.

C. Reconstruction Performance

Despite learning a disentangled representation, the generated trajectories should still look like the given training samples. Thus, we compare the final reconstruction error of TraGAN and TraVAE with the polynomial approximation used in highD [17]. The reconstruction error is the mean squared error between an input trajectory and the trajectory reconstructed by the models. The polynomial approximation is generated by fitting a polynomial to the lane change trajectory via regression as described in [17]. To reconstruct a trajectory using TraGAN, the model is used as an autoencoder as described in Section III-B. In case of TraVAE, a trajectory has simply to be passed through the network. For all three models, we calculate the mean and standard deviation of the reconstruction error on a trajectory level for a validation set. This set consists of 10% of the given lane change trajectories. The results in Table III show that the relevant lateral reconstruction error of both generative models is lower than the lateral reconstruction error of the polynomial model. Thus, the quality of the learned trajectory model is as high as or higher than the quality of the trajectory model created by an expert.

D. Disentanglement and Intuitiveness of Latent Parameters

In this section, we analyze whether the latent parameters for both TraGAN and TraVAE are intuitive and disentangled. As shown in [17], the extracted lane change trajectories are diverse because of variations including the start or end lateral position and the lane change duration, among other things. Therefore, we expect the generative models to learn these or alternative intuitive parameters. Unfortunately, no quantitative metric for disentanglement or intuitiveness exists yet. Hence, we manually set the values of the latent parameters and create plots of the resulting trajectories. For each plot, a single parameter is chosen and its value is continuously varied from -1 to 1, while the values of all other parameters remain equal to 0. Each generated trajectory is assigned an individual color that depends on the value of the varied parameter. Thus, the plot allows to visually assess whether and which attribute of the modeled lane changes a parameter represents. Ideally, a smooth color gradient is visible in the generated plot along the trajectories that correlates with a trajectory defining attribute, e.g. the lateral position after the lane change.

TABLE III: Comparison of a polynomial model, TraGAN and TraVAE for reconstructing the validation set of trajectories.

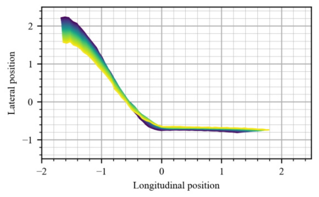
Model	Lateral MSE [m]
Polynomial model	0.055 ($\pm$ 0.05)
TraGAN	**0.0003** ($\pm$ 0.0006)
TraVAE	0.0013 ($\pm$ 0.0014)

TABLE IV: β-VAE network architecture.

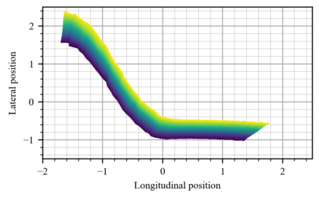
Layer	Layer Information
Input Layer	RESHAPE-(BS, SL, 1, FS)
Hidden Layer	CONV1D-(N32, K7, S3), ReLU
Hidden Layer	CONV1D-(N32, K3, S1), ReLU
Hidden Layer	CONV1D-(N32, K3, S1), ReLU
Hidden Layer	CONV1D-(N2, K3, S1)
Hidden Layer	RESHAPE-(BS, -1)
Hidden Layer	FC-(N64), ReLU
Input/Output Layer	FC-(N(LATENT DIM))
Hidden Layer	FC-(N64), ReLU
Hidden Layer	FC-(N(32·SL/FS)), ReLU
Hidden Layer	RESHAPE-(BS, SL/FS, 1, 32)
Hidden Layer	DECONV1D-(N32, K3, S1)
Hidden Layer	DECONV1D-(N64, K3, S1)
Hidden Layer	DECONV1D-(N64, K3, S2)
Hidden Layer	CONV1D-(N(FS), K3, S1)
Output Layer	RESHAPE-(BS, SL, FS)

As mentioned in Section IV-A, we have trained our TraGAN with eight latent and ten noise parameters. In Fig. 8, the four latent parameters that influence the generated trajectories the most are shown. The variation of the latent parameters show a variation of the lateral lane change start in Fig. 8a, the global lateral offset in Fig. 8b, the duration of the lane change in Fig. 8c and the lateral lane change end position in Fig. 8d. Each parameter correlates with a distinctive attribute, which proves that the TraGAN has learned a disentangled latent representation.

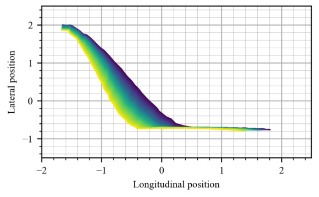
In contrast, four latent parameters were used to train the TraVAE. Fig. 9 shows the three disentangled latent parameters and the last parameter (Fig. 9d) that does not effect the generated trajectories. The parameter in Fig. 9a represents the lateral offset of the lane change. The width of the lane change is controlled by the parameter in Fig. 9b. Finally, varying the value of the parameter



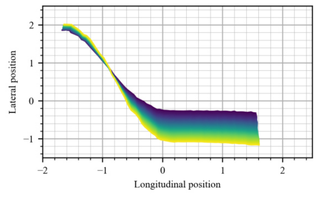
(a) Latent parameter 0.



(b) Latent parameter 3.

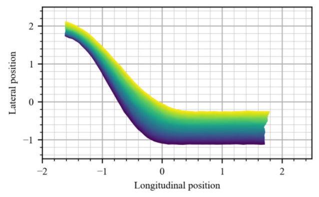


(c) Latent parameter 6.

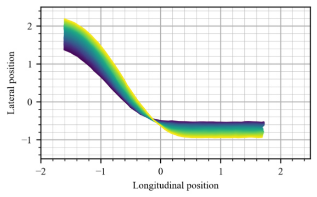


(d) Latent parameter 7.

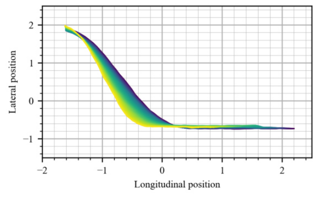
Fig. 8: Four latent parameters that have been learned by TraGAN.



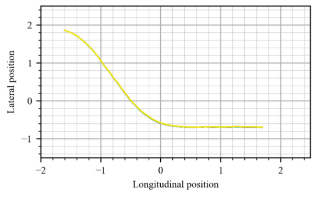
(a) Latent parameter 1.



(b) Latent parameter 2.



(c) Latent parameter 3.



(d) Latent parameter 4.

Fig. 9: Four latent parameters that have been learned by TraVAE.

shown in Fig. 9c influences the longitudinal distance. Almost similar to our results using the TraGAN, the parameters of the TraVAE each represent a distinctive attribute of the trajectories. However, the lateral start and end positions are handled by two instead of three parameters.

Both of our models are able to reconstruct trajectories and create a disentangled latent representation. The learned parameters are intuitive and similar to the parameters chosen manually by experts in [17]. However, TraGAN shows a better performance, as it has a lower reconstruction loss, and finds four disentangled and intuitive parameters. In comparison, the latent space learned by TraVAE is less disentangled and less intuitive. However, one downside of TraGAN is that training a GAN is harder than training a Variational Autoencoder due to the problems mentioned in Section II-C.2.

E. Analysis and Augmentation of the Dataset in Latent Space

Our networks allow to map a trajectory to its representation in the latent space by either using the classifier of the TraGAN or using the encoder of the TraVAE. Thus, we are able to extract the latent codes for all given trajectories in the dataset and perform analyses in the latent space. For example, we can easily analyze the frequency of certain lane changes, known as exposure, by histograms or a kernel density estimation. If the dataset is used to train a separate neural network, the exposure allows an assessment of the dataset's diversity. When testing a HAV, the exposure helps to guide the test case generation and test case evaluation. For instance, common lane change maneuvers should be included into tests, whereby the performance of the HAVs is weighted, among other criteria, by the frequency. Another use case is to analyze the location of failed test cases and to cluster those in the latent space. Thus, the localization in the latent space could help to find the cause of the problem.

After analyzing the latent space of a dataset, we can use the generative parts of the networks for augmentation. This can compensate for an unbalanced dataset where samples are underrepresented in some regions of the latent space. This in turn helps a separate neural network to work more robustly on any sample in the latent space.

Additionally, gaps in the latent space can be found by the statistical analysis and filled by the generator. These gaps represent lane changes that are not present in the dataset. In Fig. 10, a kernel density estimation based latent space analysis for TraGAN trained on the highD dataset is shown. The white areas in the plot represent gaps in the dataset. As the trained model has learned the general shape of the lane change maneuver trajectories, these gaps can be filled using the model. Despite not occurring in the training dataset, realistic trajectories have been generated for all latent codes along the dotted line. The color gradient shows the correspondence between the value of the seventh latent parameter and the generated trajectory.

V. Conclusion

In this paper, we have presented two fully unsupervised, data-driven approaches to model maneuver trajectories. We have compared specialized neural network architectures to create networks acting as accurate models, which generate new trajectories from an set of intuitive latent parameters. Both networks, called TraGAN and TraVAE, are able to generate synthetic lane change maneuver trajectories that are realistic. We have shown that the networks are able to identify and learn intuitive attributes such as the lateral start and end positions. However, TraGAN's latent space is more intuitive and disentangled. We have presented how these generative models can be used to analyze and augment a given dataset in the latent space. Thus, both generative models can be utilized to improve other neural networks or to guide the test case generation for the safety validation of HAVs. In regards to future work, we want to extend our networks to include additional types of maneuvers.

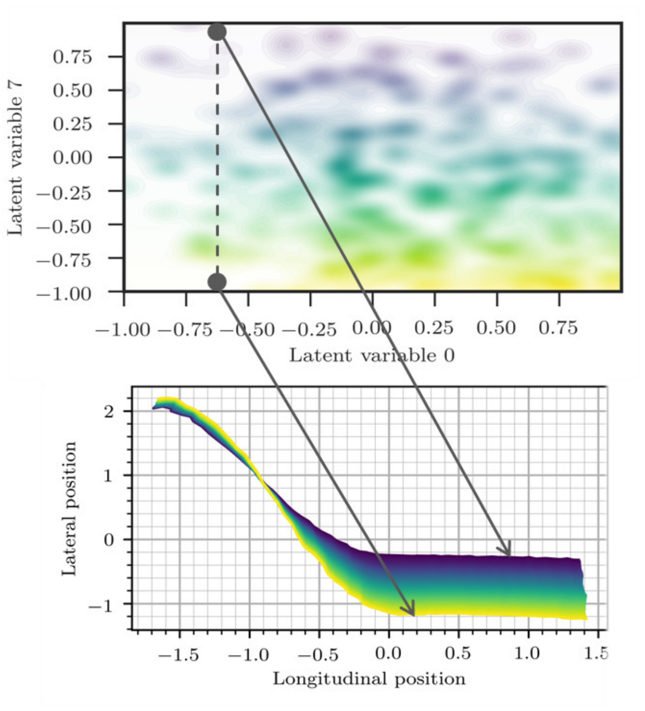


Fig. 10: Kernel density estimation for the latent parameters 0 and 7 derived by TraGAN and interpolation in the latent space for parameter 7, which represents the lateral lane change end position.

References

[1] A. Pütz, A. Zlocki, J. Küfen, J. Bock, and L. Eckstein, "Database approach for the sign-off process of highly automated vehicles," in *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, 2017.

[2] W. Wachenfeld and H. Winner, *Die Freigabe des autonomen Fahrens*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 439–464.

[3] "Pegasus project." [Online]. Available: http://www.pegasus-projekt.info/

[4] "Enable s3." [Online]. Available: https://www.enable-s3.eu/

[5] C. Amersbach and H. Winner, "Functional decomposition: An approach to reduce the approval effort for highly automated driving," in *8. Tagung Fahrerassistenz*. München: Lehrstuhl für Fahrzeugtechnik mit TÜV SÜD Akademie, 2017.

[6] "Shrp2." [Online]. Available: http://www.trb.org/StrategicHighwayResearchProgram2SHRP2/Blank2.aspx

[7] G. Bagschick, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *29th IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[8] W. Yao, H. Zhao, F. Davoine, and H. Zha, "Learning lane change trajectories from on-road driving data," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, June 2012, pp. 885–890.

[9] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems (NIPS)*, 2016, pp. 2172–2180.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS) 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[11] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 3, 2018, p. 5.

[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[13] S. Zhou, W. Zhao, J. Feng, H. Lai, Y. Pan, J. Yin, and S. Yan, "Personalized and Occupational-aware Age Progression by Generative Adversarial Networks," *CoRR*, Nov. 2017.

[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[15] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in beta-vae," in *NIPS Workshop on Learning Disentangled Representations: from Perception to Control*, 2017.

[16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Conference proceedings: papers accepted to the International Conference on Learning Representations (ICLR)*, 2014.

[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 *21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[20] M. L. Ho, "Studies on lateral control and lane changing algorithms for application in autonomous vehicles," Ph.D. dissertation, Hong Kong Polytechnic University (People's Republic of China), 2007.

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5767–5777.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems (NIPS)*, 2014, pp. 3104–3112.

[23] A. Makhzani and B. J. Frey, "Pixelgan autoencoders," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 1975–1985.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.