VeGAN: Using GANs for Augmentation in Latent Space to Improve the Semantic Segmentation of Vehicles in Images from an Aerial Perspective

Robert Krajewski
RWTH Aachen University
krajewski@ika.rwth-aachen.de

Tobias Moers
fka Aachen
tmoers@fka.de

Lutz Eckstein
RWTH Aachen University
eckstein@ika.rwth-aachen.de

Abstract

Generative Adversarial Networks (GANs) are a new network architecture capable of delivering state-of-the-art performance in generating synthetic images in various domains. We train a network called VeGAN (Vehicle Generative Adversarial Network) to generate realistic images of vehicles that look like images taken from a top-down view of an unmanned aerial vehicle (UAV). The generated images are used as additional training data for a semantic segmentation network, which precisely detects vehicles in recordings of traffic on highways. While images are commonly generated randomly for a content-based augmentation, we leverage ideas from the domain of active learning. Using a network which is based on the InfoGAN architecture allows mapping existing vehicle images to a latent space representation. After mapping the complete training dataset, we perform the augmentation in the latent space. The applied techniques include creating variations of given hard negative samples and generating samples in sparsely occupied areas of the latent space. We improve the IoU of the semantic segmentation network from 93.4% to 94.9% and reduce the mean positional error of the detected vehicles' centers from 0.51 to 0.37 pixels in longitudinal and from 0.21 to 0.17 pixels in lateral direction.

1. Introduction

A current trend in the automotive sector is the development of highly automated vehicles, which use a rich set of sensors to autonomously navigate through traffic. Using state-of-the-art sensors and algorithms, vehicles are able to drive autonomously on highways and in urban environments. Nevertheless, the safety validation of these systems is still an open issue. Since automated vehicles do not have to be tested only in critical driving scenarios, as e.g. an emergency braking assistant, but also in every other scenario, the testing effort increases enormously [24]. Current methods, which mainly focus on real world test drives, are

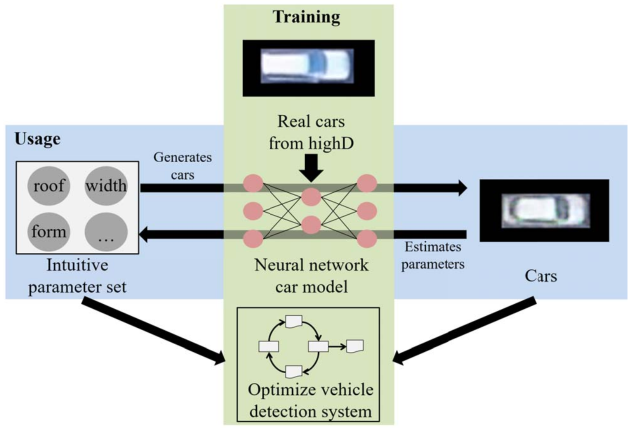

Figure 1. Illustration of the overall system. We use top view images of cars to train a GAN for generating synthetic car images from a set of learned latent parameters. In multiple iterations, we improve a semantic segmentation network by augmenting the training dataset by synthetic samples generated by the GAN.

unfeasible due to the rare occurrence of critical scenarios. To cover all relevant scenarios on motorways, more than 6 billion test kilometers would be required for safety validation [24]. Approaches to reduce the testing effort utilize simulations to accelerate the test execution, and scenario-based testing that interprets traffic as a sequence of scenarios [19]. A scenario precisely defines both the static infrastructure and the dynamic objects, including vehicles and their driven maneuvers [1]. Through this decomposition of traffic, the testing effort can be reduced as redundant tests can be avoided and more critical scenarios can be tested [26].

However, an open problem is the definition of all relevant driving scenarios. In addition to that, statistics like the occurrence probability of these scenarios are necessary to guide the simulations for safety validation like in [26]. A solution to validate the completeness of a set of scenarios and derive statistics, e.g. for a highway pilot, are datasets of vehicle trajectories on highways. Since the size and quality of existing datasets like NGSIM is limited [4], we have cre-

978-1-7281-1975-5/19/$31.00 ©2019 IEEE
DOI 10.1109/WACV.2019.00158

ated our own dataset using UAVs. The vehicle trajectories can be extracted from the recorded videos by detecting and tracking every visible vehicle. In comparison to traditional recording methods like sensor-equipped test vehicles, UAVs do not influence the traffic and do not suffer from problems like occlusion due to the aerial perspective. While the vehicle trajectories themselves were published as highD dataset [14], this paper focuses on the detection of the vehicles in the recordings.

To obtain an overall picture of the traffic on a given highway section, a high flight altitude is necessary. However, a increasing the flight altitude reduces the size of the vehicles in the recorded images and thus makes precise detections more difficult. Also, varying lighting conditions, complex shadows caused by signs and trees, and appearances of rare vehicles are typical challenges. But in order to create a high quality trajectory dataset, ideally every vehicle must be detected with pixel accuracy. Thus, a huge set of annotated images covering the described variety is necessary to create a robust detection system based on Convolutional Neural Networks. Unfortunately, creating such a training dataset is time and cost intensive because of the label accuracy required and heavy biases, e.g. in the distribution of vehicle appearances. Common methods like flipping, rotating, changing the contrast and brightness of images can be used to augment the training data and improve the robustness of the detection system. However, these methods can not be used to augment the data on a content level by e.g. generating variations of vehicle appearances to cope with content-based distribution biases.

To ease the creation of an extensive training dataset and to train a robust vehicle detection system, we utilize generative models for a content-based augmentation of training data as shown in Figure 1. Generative Adversarial Networks (GAN) [7] are the current state-of-the-art for synthesizing images and are able to generate images that are nearly indistinguishable from real samples in several problem domains. Therefore, we create and use a network called VeGAN to generate synthetic images of vehicles recorded from an aerial perspective at a high flight altitude. Additionally extending the GAN to match the InfoGAN architecture [3] allows to control specific attributes of the generated vehicle images like the color or size more easily. Further, the InfoGAN architecture allows to map an existing image to a low-dimensional latent space representation, the latent code. After mapping all images of the training dataset for the detection system to the corresponding latent code, the analysis of the detection results and hereafter the content-based data augmentation are performed in the latent space. Inspired by active learning, we actively choose points in the latent space whose corresponding images are not well handled by the detection system. After generating images from the selected latent codes using VeGAN, the synthetic images are added to the training dataset of the detection system. The detection system itself uses a semantic segmentation network and generates bounding boxes from the output of the network. Although all described methods are also applicable for trucks, we have focused our work on the generation and detection of cars.

Our main contributions are:

1. We propose a Generative Adversarial Network to generate synthetic top-view images of vehicles.
2. We analyze the controllability and interpretability of the generation process of varying sized images of vehicles using the extensions of the InfoGAN architecture.
3. We improve the reconstruction quality of the InfoGAN architecture and stabilize the training by a mean squared error loss.
4. We demonstrate the benefit of augmenting training data in latent space with the help of the VeGAN and three different sampling strategies over classical augmentation methods.

2. Related work

2.1. Generative Adversarial Networks

The research area of generative modeling has the goal to generate new realistic data samples of a given distribution. By learning the underlying distribution of the training data, the model is able to synthesize samples of the distribution. The quality of a model is evaluated by the quality of the generated synthetic samples. While previously e.g. Variational Autoencoders [10, 9] were typically used, current state-of-the-art systems in the domain of images are built on Generative Adversarial Networks, which were introduced in 2014 by Goodfellow et al. [7]. The architecture of the original GAN consists of two networks, a generator and a discriminator network. While the generator creates images from a noise vector, a discriminator distinguishes between generated and real images. By training the networks alternately, the networks play a min-max game. To improve the quality of generated images, the DCGAN architecture [20] introduces convolutional layers for both the generator and the discriminator. Currently, GANs are used for many tasks like texture synthesis [12, 2], image super-resolution [27], image inpainting [5], image editing [28] and Image-to-Image translations [29, 11].

2.2. Controllable image synthesis using GANs

Although systems based on the standard GAN architecture generate images with high quality, the generated output is hardly controllable by the input variables. Extensions of the GAN architecture allow to condition the output on the input by e.g. feeding the generator and the discriminator categorical or continuous labels during the training as presented in [18]. However, the proposed conditional GAN [18] requires labels which are not always available. In some cases, it is hardly possible to create labels, as some attributes of an image like the style can not easily be quantified. An alternative approach using unsupervised learning is the InfoGAN architecture [3], which extends the conditional GAN. A classifier network is introduced, which shares most of the layers with the discriminator and estimates the latent representation of a given image. By minimizing a mutual information loss the InfoGAN learns a disentangled representation in a completely unsupervised manner. A low-dimensional representation of an image, called latent code, is said to be disentangled, if modifying a single value of the latent code changes one specific attribute of the generated image. The vector space spanned by the elements of the latent code is called latent space.

The proposed methods in [25] allow to control the synthesis of a high resolution image semantically by feeding the generator a semantic label map. Although this allows to control the composition of the image, the appearance of each region can not be controlled intuitively in the latent space.

2.3. Active learning and content-based augmentation

Active learning is a variant of semi-supervised learning. For a pool of training samples, the labels are not given, but interactively queried by a learning algorithm. This allows to reduce the labeling effort as only selected samples have to be labeled. A typical selection strategy is the uncertainty sampling [15]. For a given model, only those samples are labeled and added to the training dataset for which the model has the least confidence about the correct output. Other strategies select samples for example by the expected error reduction [23]. With the availability of a generative model, instead of selecting existing samples for labeling, an almost indefinite number of synthetic samples can be generated. Ideally, the labels can also be automatically derived e.g. from the latent code which is fed to the generator. Thus, additional training samples and labels can be generated at no cost. In [17], a conditioned GAN is used to generate samples of underrepresented classes to improve a classification network. The proposed methods in [6] show the superiority of combining classic and GAN-based image augmentation in comparison to using only classic image augmentation methods for image classification in low-data regimes. Although these methods use GANs for the augmentation of the training data, the samples are generated randomly. To our knowledge, there is no publication on using more sophisticated selection methods for a content-based augmentation using GANs.

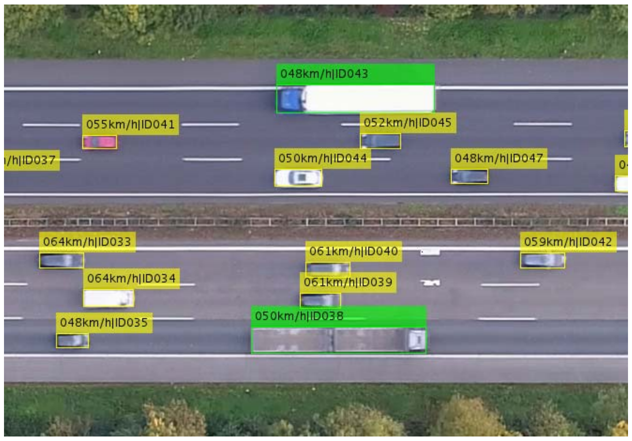

Figure 2. Sample image from the recordings under normal conditions. The recordings also include challenges like different light conditions, sign bridges and complex shadows from trees.

3. Method

3.1. Dataset

For the training of both the GAN and the semantic segmentation network, we use our own dataset of video recordings of traffic on highways. For this dataset, more than sixteen hours of traffic were recorded from an aerial perspective using a camera-equipped UAV (see Figure 2), which result in nearly one million images with in total more than 110 500 individual vehicles. The dataset contains data recorded at six measurement locations and under different lighting conditions. The orientation of the camera is always aligned with the lane markings, so that vehicles are moving horizontally and fit into non-rotated bounding boxes for labeling. We manually selected 420 non-consecutive frames across all recordings and annotated all cars using bounding boxes. The definition of a ”car” includes every vehicle with the exception of trucks and buses. Since roughly 20 cars are visible in each frame, a total of 8800 cars were accurately labeled. We split the labeled frames into a train, validation and test set (70:20:10) based on the frame’s recording time stamp. This ensured that the validation and test data set contained different vehicles at different measurement locations. To increase the amount of available training data for the GAN and the semantic segmentation networks, we augment the training samples by classic image augmentation methods, which include changing the contrast and image brightness to simulate different lightning conditions. While for the segmentation network every image was additionally flipped left/right to have each car in both driving directions, for the GAN all cars were flipped to be available in the same driving direction. Roughly 35 000 cars are used for the segmentation network and about 17 500 cars for the GAN. Finally, 1000 random patches including no cars were added as negative examples.

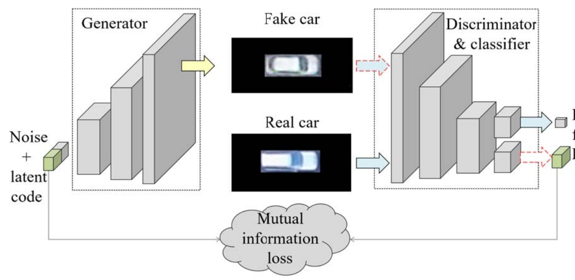

Figure 3. VeGAN architecture.

As shown in Figure 2, the size of a car in the image is very small, as the recordings were taken at a high flight altitude to get a full picture of the traffic. Thus, a single pixel corresponds to about 10x10 centimeters on the road surface. The size of the cars varies accordingly between 30 and 60 pixels in length and between 15 and 24 pixels in width. Most of the cars have a size around 48 by 19 pixels. In order to extract precise trajectories, each detection must be pixel accurate. Especially in the lateral driving direction small offsets result in a high relative error. Thus, the detection results must additionally be judged by the accuracy of the localization of the vehicle centers.

3.2. Generating car images of varying sizes

The goal of VeGAN is to generate synthetic images of all types of cars, which serve as additional training samples for the semantic segmentation network. Thus, the generator has to generate cars of varying sizes and each vehicle's size or bounding box must be known for the training of the semantic segmentation network. Since the discriminator of a GAN typically uses at least one fully connected layer for the output, it can only handle images of a fixed size. For this reason, it is not possible to create images of different sizes that only show a car. Generating image patches including a car's surroundings would allow the input images to have a fixed size. However, the bounding box of the generated car in the image patch would not be known because it cannot be precisely derived from the latent code or the synthetic image. Therefore, we chose to extract image patches including a single car in the center and pad the patches with black pixels to a fixed size (see Figure 3). This has the advantage that the location and size of generated cars can be determined easily by removing the black background. Further, the GAN only has to learn how to generate cars and the generated images of cars can be placed on any background.

3.3. VeGAN architecture

The architecture of VeGAN is based on the InfoGAN architecture [3], which adds a classifier to the discriminator network estimating the latent space representation of a given image (see Figure 3). Both the discriminator and the classifier share most of their layers but have different output layers. During training, the classifier is fed images generated by the generator network from random latent codes. By maximizing the mutual information between the input and the synthesized sample, the latent space becomes a disentangled representation of the generated images. Every element of the latent space is a latent code that the generator maps to a synthetic image of a car. In addition, each latent parameter in the latent code ideally changes only a specific attribute of the generated car.

Using the InfoGAN architecture and its classifier is beneficial for content-based augmentation. Mapping images to the corresponding latent codes allows an analysis and augmentation of the given training dataset in the latent space. Thus, variations of a false detected or rare car can be gener-

Generator architecture
CONCAT(Noise, Latent Code)
FC-(N(H*W*64/4), BN, ReLU
RESHAPE-(H/4, W/4, 64))
DECONV2D-(N64, K5, S2)
CONV2D-(N64, K5, S1), BN, ReLU
DECONV2D-(N64, K5, S2)
CONV2D-(N64, K5, S1), BN, ReLU
CONV2D-(N64, K3, S1), BN, ReLU
CONV2D-(N3, K3, S1), Sigmoid

Table 1. Generator network architecture. Note: N=Channels, K=Kernel, S=Stride, BN=BatchNorm, FC=FullyConnected, H=Height, W=Width

Discriminator & classifier architecture
CONV2D-(N64, K5, S2), BN, LReLU
CONV2D-(N64, K5, S2), BN, LReLU
DIL_CONV2D-(N64, K5, S1, D2), BN, LReLU
DIL_CONV2D-(N64, K5, S1, D2), BN, LReLU
RESHAPE-(-1))
FC-(N512), BN, LReLU
Discriminator output:
FC-(N1), SIGMOID
Classifier output:
FC-(N64), BN, LReLU
FC-(N(Latent Dim.))

Table 2. Discriminator and classifier network architecture. Note: DIL=Dilated, LReLU=Leaky ReLU, D=Dilation, Latent Dim.=Latent dimension

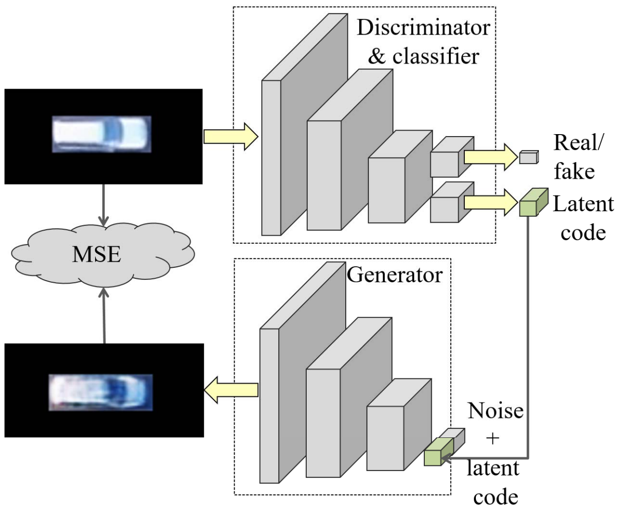

Figure 4. Using the Classifier and Generator networks as autoencoder to calculate a MSE loss.

ated from variations of its latent code. The disentanglement of the latent code allows the systematic generation of cars with underrepresented attributes. This helps the detection network to generalize, which is essential for the robustness of the detection network against fluctuations of the appearance of the cars e.g. due to lighting conditions.

As the images of the cars are smaller and hence potentially easier to generate than larger images [25], we use a relatively small convolutional network with five hidden layers for the generator (see Table 3.3) and four hidden layers for the discriminator (see Table 2). Since the training of GANs using the original adversarial loss is typically unstable, we use the improved Wasserstein GAN loss [8] which prevents the vanishment of gradients during training. In addition, we add a mean square error (MSE) loss for the training of the generator to stabilize the training and improve the quality of the reconstruction. For this loss, the classifier and generator are used as encoders and decoders like in an autoencoder. As shown in Figure 4, real images are encoded using the classifier and decoded using the generator. The loss itself is calculated between the input and the reconstructed images. Again, the classifier and a good reconstruction ability are necessary as otherwise the mapping from images to latent codes is not easily possible. An alternative solution to estimate the latent code would be by optimization. But as this is an external process, we can not use it as an additional loss to help the network converge.

3.4. Semantic segmentation network

To detect cars in aerial images, we employ a semantic segmentation network. The detection could also be done by state-of-the-art detection networks like FasterRCNN [21] as the desired output are bounding boxes. However, we found that semantic segmentation networks have less implementation overhead, are easier and faster to train, and perform better for pixel accurate detections in our tests.

To use the segmentation network for detection, it must estimate for each pixel whether it represents belongs to the background (street, greens, markings, signs) or to a car. After thresholding the classification probability, we apply blob detection on the resulting binary image and identify the biggest object to get a bounding box. We employ a fully convolutional semantic segmentation network inspired by the U-Net architecture [22]. In comparison to the original U-Net, we down sample the image only three times and limit the number of feature channels to 128. We noticed that training on complete images leads to blurry segmentations at the edges of cars. As a countermeasure, we chose to train on small patches (96x40 pixels) of the image around ground truth bounding boxes of cars as these still contain all necessary information for the segmentation network to detect a car. This also allows to train on batches of 64 patches instead of a single image per iteration as processing one full image of the highway already requires the complete memory of a GTX 1080Ti. Additionally, batch normalization can be used during the training on batches of multiple patches.

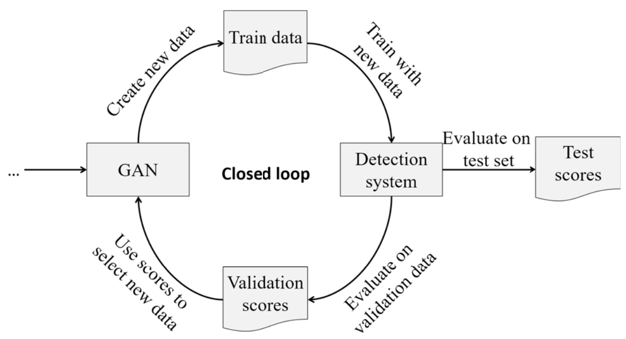

Figure 5. Improving the detection system by an iterative GAN-based augmentation process.

3.5. Augmentation in latent space

VeGAN enables the generation of additional training images including the ground truth bounding box, which the training of the detection system requires. To generate a single image, we can either sample a latent code from the latent space, let the generator generate the image of the car and finally place it on a random or selected background. However, we can also take an existing image of a car, let the classifier estimate its latent code and create variations by slightly changing the values of the latent code. Since VeGAN can potentially generate an unlimited number of synthetic images, it is important to generate only the relevant images in a controlled manner. As the training dataset for the segmentation network already includes a lot of samples, adding more of frequent or easy samples would bring no benefit. Instead, we analyze possibilities to actively select those samples in the latent space of the training samples, which potentially improve the model's detection performance the most. For this purpose we have implemented and compared several augmentation strategies. All strategies have in common that they are used iteratively (see Figure 5). After training the segmentation network with a specific training dataset, we analyze the results, generate new training samples and retrain the semantic segmentation network from scratch. Another option that we will evaluate in a future publication is to fine tune the existing network.

As baselines for the augmentation strategies, we train the network both without synthetic samples and after adding samples which are randomly sampled from the latent space and placed on random backgrounds. Generating random samples is the technique which is mostly applied in current publications for data augmentation using GANs (see Section 2.3). As one of the more elaborated methods, we create variations of those cars in the training dataset which the segmentation network couldn't segment well from the background. For our last strategy, we analyze the density of the training samples in the latent space by a kernel density estimation. We sample random points in the areas of the latent space with the lowest densities and again choose the hardest samples among them. For the last two methods, also the background causing the worst segmentation result is chosen.

4. Experiments

In this section we first present the results for training VeGAN on generating images of cars. We analyze the reconstruction performance and the disentanglement of the dimensions in the latent space. Hereafter, we show the baseline results for training the segmentation network on the original training data. Finally, we show the results for the segmentation network trained on the augmented dataset using VeGAN and different content-based augmentation strategies.

4.1. VeGAN training and results

4.1.1 Implementation Details

We use WGAN-GP [8] for stable training as the original GAN loss and LSGAN loss [16] result in mode collapse or in low image quality. Mode collapse occurs when the generator cannot generalize, but can only generate a few examples from the set of training examples. For all experiments, we use the Adam [13] optimizer with a learning rate of 5E-4 and a momentum of 0.99. The size of the generated patches is 96x40 pixels which matches the size of the training images for the segmentation network. We use the weights $\lambda_I = 5$ for the mutual information loss and $\lambda_{MSE} = 0.3$ for the MSE loss. The input for the generator is a 40-dimensional vector, which consists of 20 latent and 20 noise variables. For faster convergence, we pretrain the GAN for 5000 generator iterations by feeding the generator with random input and using random images of cars as output. Hereafter, we run the training for 35 000 iterations. The mentioned parameter values are derived from empirical tests.

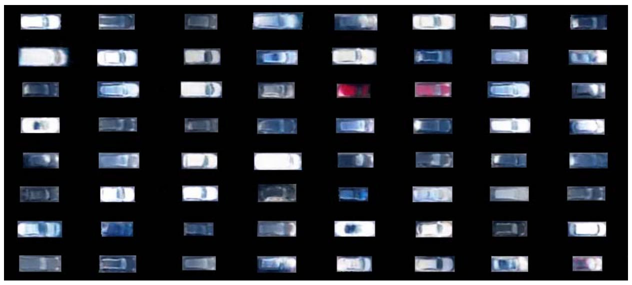
Figure 6. Cars reconstructed by VeGAN.

4.1.2 Reconstruction quality

To assess the quality of generated images, we analyze both random generated and reconstructed samples visually. Reconstructed samples are generated by applying the classifier on an existing image and feeding the generator the estimated latent code combined with a random noise vector. As shown in Figure 6, the results for most of the cars are close to the original images. However, for some of the cars, especially very colorful ones, not all details are kept. This was expected, as the details potentially have too little influence on the loss functions. We have noticed that the reconstruction quality benefits heavily from the MSE loss. Typically, if the MSE loss weight is too small, important details of the cars could not be reconstructed.

4.1.3 Latent space disentanglement

The main feature of the InfoGAN is that without using any labels, the network learns to assign the dimensions of the latent space to attributes of the cars in the image space. Thus, changing the value of a single element in the latent code ideally changes a single attribute of the generated car. The results show, that this works for multiple attributes. For example, the network learns to generate cars of varying sizes. The size of the cars is encoded in a single latent parameter. Another attribute is the color of the car (see Figure 7). As most of the cars recorded at highways are perceived as black, white or red, using categorical instead of continuous latent parameters could be beneficial for generating those. Nevertheless, the network learns that the color of the cars is a specific attribute which is captured by a latent parame-

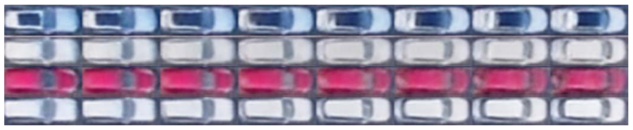
Figure 7. Latent parameter controlling the car roof's position. Please note that in this and other figures the black border around the car has been removed for clarity.

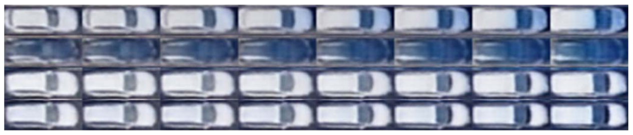
Figure 8. Latent parameter controlling the tilting.

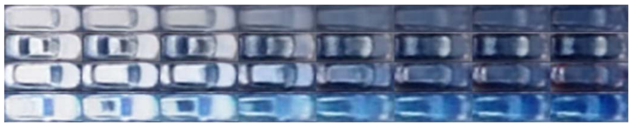
Figure 9. Latent parameter controlling the color.

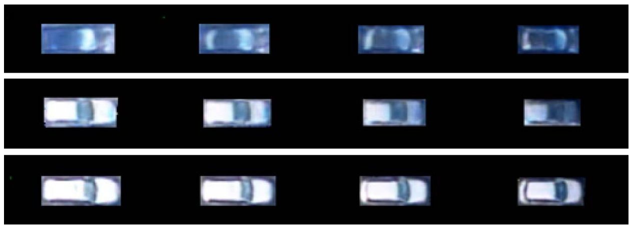
Figure 10. Examples for changing a car's size.

ter. Two other parameters clearly describe the perspective of the recorded car. While one parameter describes the tilting caused by the varying vertical distance to the image center (see Figure 8), the other parameter describe the panning due to the horizontal distance to the image center (see Figure 9). However, other parameters don't seem to have a consistent effect on the generated images. Our assumption is that this is caused by the correlation of some attributes to other attributes or car type. Since the training dataset contains limousines, transporters and other types, the introduction of separate generators or categorical parameters could further improve the disentanglement.

4.1.4 Generating car images of varying sizes

As already described, the generator learns, unsupervised, to encode the size of the generated cars in a single latent parameter (see Figure 10). We also analyzed if the size of the reconstructed cars matches the size of the original cars. The mean squared error measured by the movements of the edges is less than a single pixel.

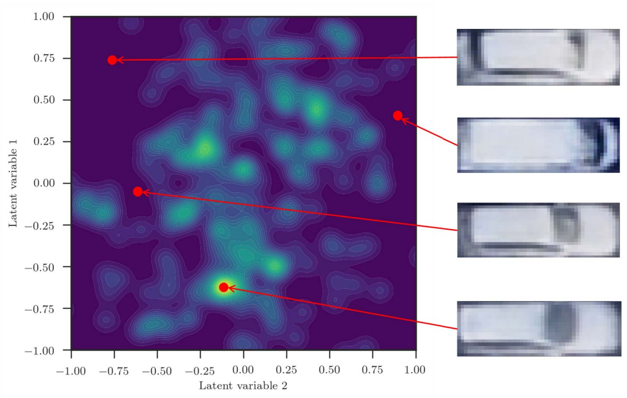

Figure 11. Density plot of the latent representations of the training data. The generator also generates realistic looking samples in sparsely populated areas.

4.2. Analysis in latent space

For the analysis in the latent space, each training sample is transformed into its latent representation by applying the classifier on it. For visualization purposes, we apply a kernel-density estimation and plot the density values for only two dimensions simultaneously. As shown in Figure 11, the training samples are not distributed equally in the latent space and gaps exist. However, the generator is able to generate realistic samples from latent codes in these gaps and thus can generate images of cars, which are not part of the training dataset.

4.3. Semantic segmentation network

4.3.1 Implementation details

Before we augment the training dataset using VeGAN, we train the semantic segmentation network on the extracted real image patches for a baseline performance. We choose an ADAM optimizer with a learning rate of 1E-4 and a momentum of 0.99. The size of the patches is 96x40 pixels and a batch consists of 64 patches. We train the network 50 epochs on 35 000 samples, which makes the network converge based on empirical evidence.

4.3.2 Baseline results

To evaluate the performance of the segmentation network, we choose pixel-based and bounding-box-based metrics. To visualize the pixel-based true positive rate in comparison to the false positive rate of segmented pixels at different thresholds we use ROC curves. The first bounding-box-based metric is the intersection over union (IoU) of the detected car's bounding boxes, which is a common choice to evaluate segmentation networks. As second metric we use a more specialized metric for our use case. Our goal is to extract the trajectory of the cars over time and derive accu-

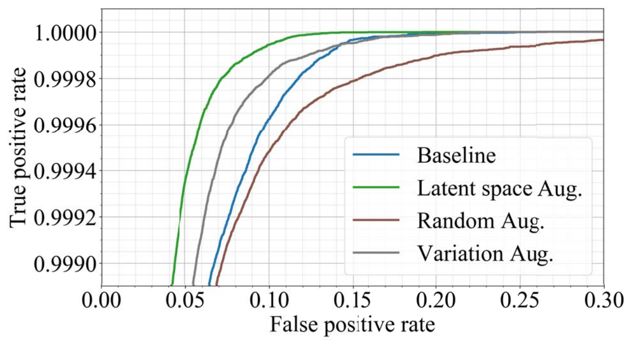

Figure 12. ROC curves for the resulting performance of the segmentation network depending on the augmentation strategy to create the training dataset.

rate values for the velocity and acceleration. As we track each car by the center of its bounding box, the center position error, measured in pixels, should be minimal. We chose to consider the error in longitudinal (CP_x) and lateral (CP_y) driving direction individually as the vehicle speeds and distances in both directions are on a different scale. For both bounding-box-based metrics, the segmentation images are binarized using a threshold of 0.9 (0: background, 1: car). For the baseline training dataset using classic augmentation methods, the IoU is 0.934 while CP_x and CP_y are 0.51 pixels and 0.206 pixels (see Table 3). Comparing samples from the test dataset shows that the network especially has problems with cars appearing similar to the background due to bad lighting and dynamic shadows (see Table 4). Neither increasing the number of epochs nor the number of training samples using classic augmentations improved the results.

4.4. Augmentation in latent space

To evaluate the benefit of VeGAN's generated car images, we ran a set of experiments. As a baseline, we train the semantic segmentation network using classic image augmentation methods. As comparison, we ran a set of three experiments using additionally samples generated using VeGAN as described in Section 3.5. For every experiment except for the baseline, the network is trained multiple times from scratch with the parameters described in Section 4.3.1.

Table 3 shows that every content-based augmentation

Method	IoU	CP_x	CP_y
Baseline	.934	.51	.206
Random	.948	.39	.184
Variation	**.949**	**.37**	**.169**
Latent space	.948	**.37**	.174

Table 3. Resulting performance of the segmentation network on the test dataset.

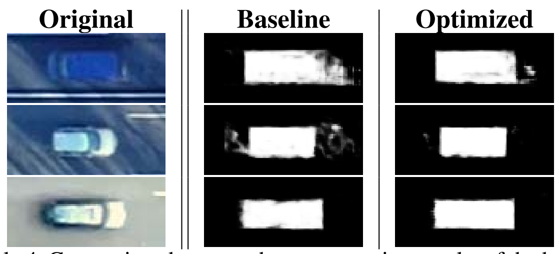

Table 4. Comparison between the segmentation results of the baseline and latent space augmentation strategy on test images.

strategy using VeGAN is superior to classical augmentation methods as measured against the test set. Every augmentation strategy increases the Intersection-over-Union and reduces the error of the bounding box center positions for both directions. Especially for reducing the lateral error of the center positions, the augmentation strategies creating variations of existing cars (Variation Aug.) and the gap filling in the latent space (Latent space Aug.) perform better than the random augmentation. Comparing the ROC curves in Figure 12, the latent space augmentation performs best. The latent space augmentation strategy has the lowest false positive rate, which results in less pixels in shadows being classified as cars as shown in Table 4. Also the segmentation at the boundary between the cars and the background is clearer, resulting in improved center position errors.

5. Conclusion

In this work, we improve the accuracy of a semantic segmentation network for detecting cars in aerial images using synthetic images. In addition to classic augmentation methods, we train a generative adversarial network based on the InfoGAN architecture, called VeGAN. The created network can both generate synthetic images of cars from a random point in the latent space and map images of cars to this latent space. Although no labels were provided, the network learns to assign the latent parameters attributes like the panning, tilting, the size or the color of the cars in given image patches. We employ the trained VeGAN for a content-based augmentation of the training data in the latent space using strategies inspired by active learning. The results show that the latent space augmentation using VeGAN and the reconstruction strategy is superior to classic augmentation methods as new variations of cars can be generated. Using the variation strategy, the performance of the segmentation network improves from 93.4% to 94.9% IoU and positional errors decrease from 0.51 to 0.37 pixels in longitudinal and from 0.206 to 0.169 pixels in lateral driving direction. Our next steps include introducing categorical latent parameters and training a generator for background images to augment the training data in a joint latent space.

References

[1] 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.

[2] U. Bergmann, N. Jetchev, and R. Vollgraf. Learning texture manifolds with the periodic spatial GAN. *CoRR*, abs/1705.06566, 2017.

[3] 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)*, pages 2172–2180, 2016.

[4] B. Coifman and L. Li. A critical evaluation of the next generation simulation (ngsim) vehicle trajectory dataset. *Transportation Research Part B: Methodological*, 105:362–377, 2017.

[5] U. Demir and G. B. Ünal. Patch-based image inpainting with generative adversarial networks. *CoRR*, abs/1803.07422, 2018.

[6] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. Synthetic data augmentation using gan for improved liver lesion classification, 2018.

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

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[9] 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. *CoRR*, 2016.

[10] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141, March 2017.

[11] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, July 2017.

[12] N. Jetchev, U. Bergmann, and R. Vollgraf. Texture synthesis with spatial generative adversarial networks. *CoRR*, abs/1611.08207, 2016.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] 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 *2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[15] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994.

[16] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang. Multiclass generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

[17] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi. Bagan: Data augmentation with balancing gan, 2018.

[18] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[19] 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.

[20] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[22] O. "Ronneberger, P. Fischer, e. N. Brox, Thomas", J. Hornegger, W. M. Wells, and A. F. Frangi. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[23] N. Roy and A. McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448, 2001.

[24] W. Wachenfeld and H. Winner. The release of autonomous vehicles. In *Autonomous Driving*, pages 425–449. Springer Berlin Heidelberg, 2016.

[25] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[26] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan. Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):595–607, March 2017.

[27] L. Zhao, J. Liang, H. Bai, A. Wang, and Y. Zhao. Simultaneously color-depth super-resolution with conditional generative adversarial network. *CoRR*, abs/1708.09105, 2017.

[28] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. *CoRR*, abs/1609.03552, 2016.

[29] 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.