Autonomous driving scenario generation using Generative Adversarial Networks

Preprint · November 2020

CITATIONS
0

READS
959

3 authors:

Mallek Mziou Sallami
Atomic Energy and Alternative Energies Commission
63 PUBLICATIONS **166** CITATIONS
SEE PROFILE

Samy Kerboua-Benlarbi
Sorbonne Université
10 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Abdelkrim Doufene
Massachusetts Institute of Technology
44 PUBLICATIONS **133** CITATIONS
SEE PROFILE

Neurocomputing

Autonomous driving scenario generation using Generative Adversarial Networks

--Manuscript Draft--

Manuscript Number:	
Article Type:	Full Length Article (NN)
Keywords:	Autonomous driving, Time Series, Deep Learning, Learning systems, Engineering and Applications
Corresponding Author:	Mallek Mziou Sallami, Ph. D. FRANCE
First Author:	Samy Kerboua-Benlarbi
Order of Authors:	Samy Kerboua-Benlarbi
	Mallek Mziou Sallami, Ph. D.
	Abdelkrim Doufene
Abstract:	Machine Learning and its related subjects are increasingly enabling the development of autonomous driving systems. One common obstacle comes from the need in various driving scenarios to train or evaluate models. Indeed, an adequate coverage of the situations from which a self driving system is performing will guarantee its efficiency in terms of safety and robustness. Adversarial models have been widely used for automatic generation, with several recent works designing models taking into account the real data manifold coverage. In this work, we propose a definition of what should be a driving scenario and a simple representation of the latter that enables the use of image-based generative adversarial networks. We conduct an empirical study meant to serve as a recommendation protocol towards autonomous driving scenario generation. Furthermore, we assess the feasibility of our method, which reaches a coverage of the manifold on which our data is theoretically lying for each set of networks used in our experiments.

Abstract

Autonomous driving scenario generation using Generative Adversarial Networks

Samy Kerboua-Benlarbi[a], Mallek Mziou-Sallami[a,b,*], Abdelkrim Doufene[a]

[a]*Institut de Recherche Technologique SystemX, France*
[b]*Le Commissariat à l'énergie atomique et aux énergies alternatives (CEA), France*

Abstract

Machine Learning and its related subjects are increasingly enabling the development of autonomous driving systems. One common obstacle comes from the need in various driving scenarios to train or evaluate models. Indeed, an adequate coverage of the situations from which a self driving system is performing will guarantee its efficiency in terms of safety and robustness. Adversarial models have been widely used for automatic generation, with several recent works designing models taking into account the real data manifold coverage. In this work, we propose a definition of what should be a driving scenario and a simple representation of the latter that enables the use of image-based generative adversarial networks. We conduct an empirical study meant to serve as a recommendation protocol towards autonomous driving scenario generation. Furthermore, we assess the feasibility of our method, which reaches a coverage of the manifold on which our data is theoretically lying for each set of networks used in our experiments.

Keywords: Autonomous driving, Time Series, Deep Learning, Learning

*Corresponding author
Email address: mallek.mziou@cea.fr (Mallek Mziou-Sallami)

Highlights

Autonomous driving scenario generation using Generative Adversarial Networks

Samy Kerboua-Benlarbi, Mallek Mziou-Sallami, Abdelkrim Doufene

- Adapt scenarios to the use of image-based generative adversarial networks
- Train several models either for generation and coverage purposes
- Formalize the evaluation metric to the case of driving scenarios

Autonomous driving scenario generation using Generative Adversarial Networks

Samy Kerboua-Benlarbi[a], Mallek Mziou-Sallami[a,b,*], Abdelkrim Doufene[a]

[a]*Institut de Recherche Technologique SystemX, France*
[b]*Le Commissariat à l'énergie atomique et aux énergies alternatives (CEA), France*

Abstract

Machine Learning and its related subjects are increasingly enabling the development of autonomous driving systems. One common obstacle comes from the need in various driving scenarios to train or evaluate models. Indeed, an adequate coverage of the situations from which a self driving system is performing will guarantee its efficiency in terms of safety and robustness. Adversarial models have been widely used for automatic generation, with several recent works designing models taking into account the real data manifold coverage. In this work, we propose a definition of what should be a driving scenario and a simple representation of the latter that enables the use of image-based generative adversarial networks. We conduct an empirical study meant to serve as a recommendation protocol towards autonomous driving scenario generation. Furthermore, we assess the feasibility of our method, which reaches a coverage of the manifold on which our data is theoretically lying for each set of networks used in our experiments.

Keywords: Autonomous driving, Time Series, Deep Learning, Learning

*Corresponding author
Email address: mallek.mziou@cea.fr (Mallek Mziou-Sallami)

1. Introduction

Following the advancements of machine learning and especially deep learning, autonomous vehicles need various and numerous data in order to capture interesting behaviors, or patterns in a more general way, that will help them drive safely and efficiently. A first bottleneck is to be found in the fact that these data have to describe as many situations as possible, to ensure the safety of self driving vehicles. Nevertheless, it's practically impossible to gather all the situations that could occur on a road. In order to overstep this problem, one may generate new scenarios, e.g a situation in which a driving system would be engaged. In fact, the choice of a generation method directly depends on what definition could be used to define a scenario. In this work, a scenario will be considered as a sequence of elements that constitute a description of the environment depicted around the autonomous vehicle. Our work and future ones will benefit from it as it is both precise and general enough to consider all possible models and all possible learning paradigms. Indeed, a model would use such depiction as inputs for training, evaluation or even certification at the same time. For example, an autonomous driving scenario could be several numerical sequences, each describing a parameter of the environment at each time step, such as speed or steering. Here, it is important to notice that a scenario is not a function of the common perception that is usually described for autonomous systems inputs (cameras, ...). Moreover, we have to take into account the fact that autonomous driving data, regardless of how they were gathered, are time-dependent. Therefore,

it is appropriate to consider them as time series. A problem arises when it comes to process this representation shared by all scenarios, in terms of space. This is called the coverage problem. The search for a representation that effectively defines all our scenarios and helps them maintain the meaning of the situations they describe. A well-defined coverage would imply an understanding of the relevant data and an effective way to generate them. Thus, the notion of manifold is appropriate because we theoretically acknowledge that a high dimensional data distribution, such as the autonomous driving scenarios one, is lying on a lower dimension manifold [1]. Computing the real shape of a manifold is not trivial, but we can model its behavior thanks to a specific family of neural networks frameworks, called generative adversarial networks. So, the objective is to use a model from the former and generate data implicitly along the manifold, on which every realistic data points, scenarios in our case, are closely lying. Once a generation process is build, a proper evaluation of its capacities needs to be done. Topological Data Analysis and persistent homology will then be used to identify quantitatively how good a reached coverage is.

Using these ideas, our contribution is composed of several steps:

- Adapt scenarios to the use of image-based generative adversarial networks
- Train several models either for generation and coverage purposes
- Formalize the evaluation metric to the case of driving scenarios

The remaining of this paper is organized as follows. In section 2, a brief survey of Generative adversarial networks is given with a focus on time series

generation and evaluation metrics. After that, we will present our approach for scenarios processing and time series conversion to image. The fourth section is devoted to experimental setup. In section 5, we will analyze our generation results independently, first in terms of learning dynamics from as far as stability and equilibrium are concerned, and then in terms of driving data generation in a quantitative and qualitative manner. Finally, we conclude and present some perspectives for future work in Section 6.

2. Related work

2.1. Generative adversarial networks

Generative adversarial networks or *GANs* [2] emerges from game theory. Indeed, they're composed of two entities: a generator G and a discriminator D. The former tries to fool D by generating realistic samples, while the latter always tries to be able to distinguish G's creations from real examples (Figure 1). Given $\mathbb{P}_r$ the real data distribution and $\mathbb{P}_g$ the generated one, this *min-*

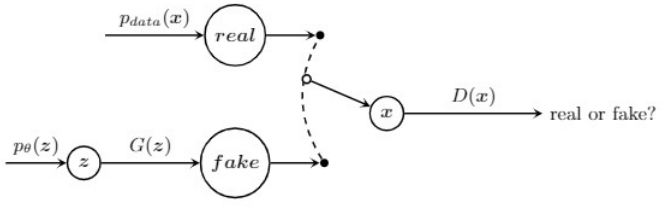

Figure 1: The adversarial framework for data generation

max game attempts to find a Nash Equilibrium by alternated optimization, given the below expression as a cost function:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r}[\log(D(x))] + \mathbb{E}_{z\sim\mathbb{P}_g}[\log(1 - D(z))]$$

Another formulation was introduced to overcome gradient issues arising from the initial one. The generator's update rule then went from $\mathbb{E}_{z\sim\mathbb{P}_g}[\log(1 - D(z))]$ to $-\mathbb{E}_{z\sim\mathbb{P}_g}[\log(D(z))]$. Nevertheless, numerous problems remain [3], such as the *mode collapse* or the theoretical assured existence of a "perfect" discriminator which could separate the supports of the two data distributions. Several works brought methods to improve the learning process of GANs [4, 5]. A big enhancement was made with the use of the Wasserstein distance which gave the Wasserstein GANs or *WGANs* [6], thanks to the Kantorovich-Rubinstein duality [7]. The cost function is then similar to the one observable in vanilla GANs, except that we do not talk about D as a discriminator anymore, but as a "critic". In fact, a WGAN does not minimize a Jensen-Shannon divergence [2], but learns a real function instead. It is worth noticing that it is often needed to iterate several times over the discriminator before iterating over the generator, in order to ensure an efficient sharing of information between the two parts of the framework. It holds to the below expression as a cost function, with $\mathbb{D}$ the set of all 1-Lipschitz functions and $\mathbb{P}_g$ the generated distribution:

$$\min_G \max_{D\in\mathbb{D}} \mathbb{E}_{x\sim\mathbb{P}_r}[\log(D(x))] + \mathbb{E}_{z\sim\mathbb{P}_g}[\log(1 - D(z))]$$

To make WGANs operational, we need to enforce 1-Lipschitz continuity on the discriminator. Consequently, the first works clip the gradients at each learning iteration. Yet, this process is considered hazardous, which steered research towards the development of a gradient penalty [8]. Indeed, this penalty allows the enforcement of the 1-Lipschitz continuity by interpolating points on lines between the real data distribution and the generated data one. However, this term takes in account really specific examples, leaving

several zones of the real data manifold unexplored. In addition, interpolated data might be far from both manifolds during the first learning iterations and nothing insures that the two distributions will come up to each other enough so that the 1-Lipschitz continuity exists. A consistency term [9] was proposed, as an extension of the gradient penalty discussed above. This term leads to a better exploration of the real data manifold, and enforces Lipschitz continuity to the real data surroundings: these regularizations theoretically improve the stability and convergence of GANs, and also the generation itself. It is worth notice that gradient penalty has not been used only for WGANs [10].

There are many different cost functions for many different models including Bayesian formulations [11] or a third network [12], using a genetic approach [13] or even using Variational auto-encoders [14]. However, a recent work [15] has shown that the choice of a cost function does not matter compared to optimizing hyper-parameters. Empirical studies have validated these elements and gone further by showing that beyond the choice of model parameters, regularization and normalization play a real role in the stability of learning and the quality of the images generated [16]. By regularization, we mean any constraint placed on the cost function or any formulation that regulates the standard of gradients [8, 9, 17, 18]; normalization concerns the weights of the network [19, 20, 21]. Other studies have brought a new formalization of GANs to show their difficulty in converging regardless of the chosen cost function [22], or have precisely proven interesting convergence properties by applying the gradient penalty to a classic GAN [10], but also by analyzing the gradient descent in GANs [23]. Several very recent models have surpassed all

the others thanks to the attention mechanism [24] or by scaling GANs [25].

2.2. GANs with time series

From recurrent neural networks and its enhancements, sequences can then be generated: indeed, since the prediction of the next action is at the heart of these networks, it is possible in the case of natural language for example, to sample the beginning of a sentence and to ensure that words are predicted one by one. However, this task is not trivial and can lead to disappointing results [26]. It would then be interesting to consider GAN as a "framework" before being a model: by this we mean that a GAN consists of a generator and a discriminator which can be convolution networks, but theoretically any other type of architecture, continuing whatever it manages to explore the said manifolds. With the help of previous recurrent networks, it is possible to design GANs capable of generating sequences, taking care to adapt the principle of recurrence to the cost functions stated so far. The non-differentiability of the choice of each word after the softmax poses nevertheless a problem with regard to the backpropagation required to train the discriminator. Several studies have tried to overcome this problem, using methods from reinforcement learning [27], using a different formulation of the softmax function [28], or using the capabilities of auto-encoders to sample not words but entire sentences on the explored manifold [29]. Some work, particularly in the generation of continuous sequences (music, medical field,...) attest the possibility of using recurrent networks as they stand, by using linear layers at the output of a recurrent layer to be able to apply a softmax more easily [30, 31].

Thus, RNNs and other variants can be applied to GANs, but it should be

noted that it is currently theoretically impossible to use wasserstein GANs directly [31], due to the latter's formulation (the conditions on the formulation of the Critic no longer work). We will also note the use of recurrent GANs in word sequence generation without pre-training [32] which is an interesting idea, since the discriminator in these models generally tends to become too strong too quickly [30]. The handling of missing values in time series also uses GANs on some applications [33] and there exist several recent works in the use of GANs for time series modeling [34], financial strategies fine-tuning [35]. A recent work [36] uses temporal convolutional networks in GANs for time series generation.

2.3. Evaluation of GANs

Many metrics exist for the evaluation of GANs. More specifically, since most of the work concerns the generation of natural images, the metrics seek to attest to both the quality of the images generated and their diversity. For example, we can note the *Inception Score* (IS) [4] which aims to verify if the images are varied and if they look like something real using a pre-trained classifier on a natural image dataset. The *Fréchet Inception Distance* [10] aims to improve the Inception Score by calculating the distance between two multivariate Gaussian configured by the outputs of the Inception Network on real and generated data. Other methods use a trained auxiliary classifier to distinguish generated images from false ones [37, 38]. We can also train a network on real data and measure its performance on generated data, or the opposite [31].

Other methods do not involve training from another network, such as Maximum Mean Discrepancy (MMD) [39] which assesses the dissimilarity between

two data distributions. Some variants exist, using kernels [40]. One should notice that the discrepancies that GANs use can also be used as metrics [41]. Choosing a metric depends on what you want to measure, because there are no metrics that can detect all the phenomena inherent in the generation. The characteristics most often sought are over-fitting, the quality of the data generated or sensitivity to spatial distortions. Some works list all the metrics with their strengths and weaknesses [42], while others try to understand which ones not to use, such as Parzen estimators or log-likelihood on GANs. [43]. There is still a need in a metric that can approximate the intrinsic geometry of the data to think in terms of coverage.

2.4. Evaluation by simplicial homology

Simplicial homology establishes its origins in topological data analysis. Its strength lies in the possibility of analyzing multidimensional data, which corresponds perfectly to our case. Topological data analysis or *TDA* seeks to understand the structure and the geometric nature of data. In this sense, it is able to capture invariant characteristics that reflect the reality of datasets. The principle of persistent homology is then an algebraic method to extract information about the overall structure of data, such as "holes", that could indicate in which areas they are concentrated. In other words, it makes it possible to know the areas of space where little or no data is present. This approximation is done by connecting each group of close points to represent them as multidimensional graphs called *simplicial complexes*. The filtration operation allows to build a set of simplicial complexes, by adjusting an ϵ meant to be the radius of each ball produced around each sample. Let's assume $\mathbb{X}$ a set and a metric d. The different filters are based on the use

of d to test whether several points belong to the same component. For each value that ϵ can take, we define the homology of the scatter plot on which we work, where the number of holes evolves according to the evolution of ϵ. Once filtration is obtained, persistent homology can be used to attest to the "lifetime" of observed holes: it is the calculation of the *persistence bars* [44]. Thus, we have a measure of the evolution of the approximate shape of manifolds (Figure 2). There are several ways to build simple complexes

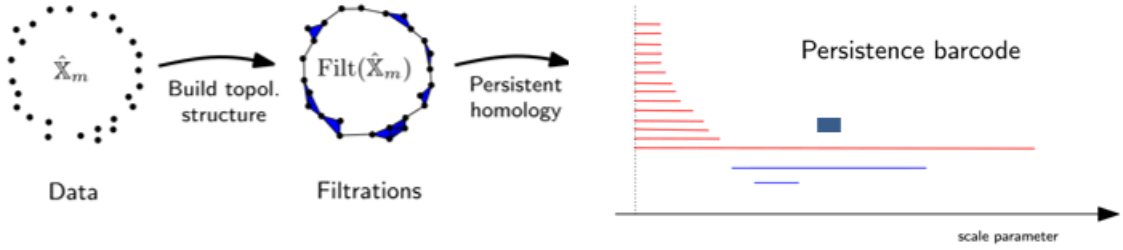

Figure 2: "Pipeline" for the study of simplicial complexes

such as *witness filtration* or *Rips Filtration*. The *Geometry score* [45] is a metric that uses results of persistent homology and more specifically persistence barcodes. By looking at the statistics on its components, we can know how long each of them lived over the period of the topological study: by this, we mean that each number of holes is weighed by the maximum value of ϵ, which gives the relative lifetime of this number of holes. This is a confidence measure since a number of holes that lasts for a while (in terms of ϵ evolution) indicates that this number reflects the shape of the data manifold. Its computation requires repeating the experiment several times, and the values obtained for each relative lifetime are averaged, giving what is call their *Mean Relative Living Time*, or $MRLT$. For two datasets X and Z,

the geometry score is then calculated using the quadratic error between the two $MRLTs$ of each distribution. So the lower the geometry score is, the more it reflects the proximity to the manifold. Indeed, this would mean that the number of holes does not decrease. In short, this metric and the TDA make it possible to evaluate GANs in terms of coverage and model collapse, regardless of the size and dimension of the data used.

3. Approach

3.1. Image-based generation

The definition of a scenario implies taking into account the environment around the autonomous vehicle called *ego* over a certain period of time; therefore, our data are time-dependent signals. It is a succession of points, multivariate series (as many variables as the input of the model), in a space of several dimensions. For example, a point would characterize the curvature of the road, the speed and the steering angle of the vehicle. Each point is dependent on previous ones, as it could be in the case of trajectories, or financial values. We therefore need to build adversarial generative models that can handle temporal sequences, multivariate in our case.

Using GANs with a representation that would be obtained by converting our data into images was therefore a first step. There are many types of conversions, but the choice was made for a method using a simple discretization inspired by a recent study on the generation of univariate periodic series, but adapted to our more complex case. By doing so, we would then be able to use proven methods on image generation, and to use these methods to carry out an empirical study.

3.2. Scenarios processing

Once each component of scenarios were known as univariate time series, it was therefore necessary to consider a modelling that could be appropriate for the use of GANs. The theoretical foundations of some transformations [46] could lead to a loss of information. This problem would have made it tedious to achieve the objectives of this work. Following the idea of a fine discretization of periodic univariate series to grayscale images [47], it was decided to adapt this method to aperiodic multivariate ones. The process is not afflicted by the absence of periodicity, and it raises a question on GANs capacity to handle even more complex signals. Usually, images are encoded on 3 channels (RGB) to represent the color spectrum. Some representations exist where more channels can be reached (multiple satellites,...). It is interesting to consider one channel of a future image for each univariate series of a scenario. Computing the average over all channels and converting the resulting values into a single greyscale image was not an option. Indeed, this would not only imply a huge loss of information across all sequences of the scenario, but also destabilize the reversal process. Instead, a transformation of each univariate serie into an image is performed, and each representation is concatenated to the others, to form a multi-channel image. For example, if a driving scenario is composed of four different series (speed, road orientation,...), each of them would be transformed into a greyscale image that would then constitute a channel in our final representation, leading to a 4-channel image. On the other hand, RGB can be used, which then gives a 12-channel image.

Series discretization corresponds to a division of the ordinate axis into 2^{24}

intervals, between the minimum and maximum values of each series over the entire dataset to prevent bias, assigning to each point the index of the interval in which it's lying, i.e. a value ranging from 0 to 16777215, which is converted to 24 bits binary integers. The previous operation, called *Quantization*, then allows building triplets of values t, where $\forall v \in t, v \in 0, 256$. This is done by shifting and isolating the 24 bits in groups of 8. The last step of the conversion is a simple *resizing* of the encoded series to images of size 64×64.

Figure 3 and algorithm 1 summarize the process for the RGB representation.

Algorithm 1 Encode a driving scenario

procedure PROCEDURE ENCODE_SCENARIO
Input: $S \in [A, B]^k$ A, B min and max for each feature, $k \in \mathbb{N}^*$; $n \geq 8$ bits
    S_T : List
    for $f \in S$ **do**
        Discretize $[A, B]$ in 2^n intervals I_t, $t \in [0, 2^n]$
        Quantify : $\forall i \in f, i = t$ where $i \in I_t$
        if $n > 8$ **then**
            Cut in groups of 8 bits to obtain a value for each channel
        Reshape each remained vector V_f as a squared image
        S_T : Append V_f
    Return S_T

One shall convert scenarios to different image sizes by using another sampling rate, the duration of scenarios enforcing a specific size. Images gen-

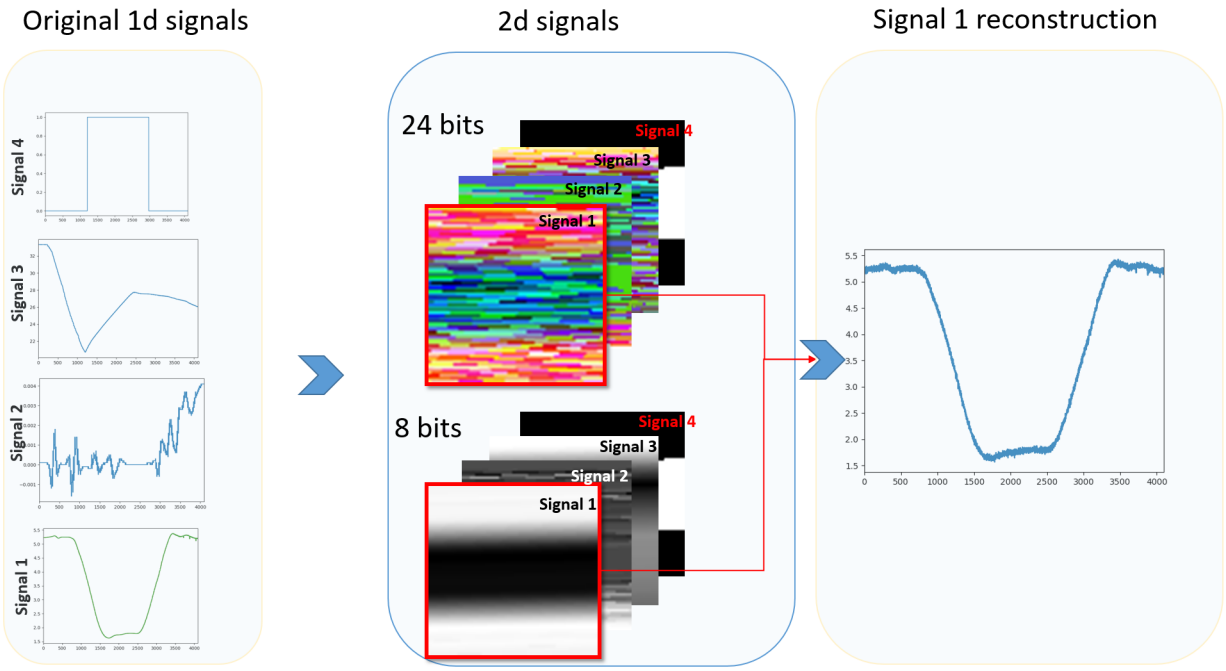

Figure 3: Complete process of time series to image conversion

erated by GANs are usually noisy. Removing noise from generated images as it stands would cause us to lose information. So, the images were first reconstructed in series, and denoised using a low-pass filter similar to [47].

3.3. Data collection

All the experiments were conducted using data from a driving simulator provided by the project partner *Groupe PSA*. The latter was designed to offer a working tool for anyone interested in the theme of autonomous cars, since it is a realistic simulator including data collection tools and a well provided native API. The simulator was then able to collect data in line with our definition of a scenario in a supervised context. Then, a study

of the generation of these scenarios was possible. The frequency at which the data were collected would induce univariate series of the same length. In our simulator, the data recovery frequency was about 256 Hz, over 16-seconds scenarios, with 4096 recovered values ($4096 = 64 \times 64$). Because of background functions calls in the simulations, this frequency rate was not fixed and small errors intervals were observed, even if it is not a real issue. All in all, 4096 values were needed to create an image of size 64×64, and 5375 scenarios were then gathered for training.

4. Experimental setup

An empirical study was conducted to evaluate the generation of driving scenarios. A DCGAN architecture was maintained throughout the study to allow an effective comparison of cost functions, but also because it is the architecture known to be the best at the image generation level (CNN for discriminator D and generator G), better than multi-layer perceptrons. The trained models correspond to the GANs and WGANs, to which the same regularization procedures with specific harshness (gradient penalty and consistency term) and the same forms of normalization (batch, layer, spectral) were applied. Based on theoretical explorations, normalizations were applied either to both networks, or put at a specific position in the case of the spectral one. The ADAM optimizer was the one of all models except the WGAN which uses RMSProp, and all hyperparameters are described in tables [1,2]. All inputs are normalized between -1 and 1 to prevent gradient problems. A nomenclature was designed to effectively identify all 108 models for the future. The latter corresponds to a sequence of several acronyms:

1. the type of GAN, i.e. 6 possible acronyms (GAN, GANGP for *Gradient Penalty*, GANGPCT for *Gradient Penalty* and *Consistency term*, WGAN, WGANGP, WGANGPCT)
2. the type of normalization (if dropout, always at 0.2):
    - **R** for a normalisation batch in D and G, **Rd** for a dropout addition in D
    - **LN** for a normalisation layer in D and G, **LnD** for a dropout addition in D
    - **RSN** for spectral normalisation only in D (batch normalisation in G), **RSnD** for dropout addition in D
    - **SN** for a spectral normalisation in D and G, **SnD** for a dropout addition in D, **SnDB** for a batch normalisation addition after the spectral normalisation in G [24])
3. **FC** for "Fully convolutional"
4. **2D** for the image data format
5. **TTUR** if two times update rule added (two different learning steps for D and G optimizers)

The hyperparameters defined for each model are described in these two tables. α and β correspond to the parameters of the generator and the discriminator/critic optimizers . n_{disc_iters} is represented as n:1 where n is the number of iterations of the discriminator before updating the generator.λ_1 is the parameter of the gradient penalty, λ_2 and M are those of the consistency term.

	GAN	GAN-GP	GAN-GP-CT	WGAN	WGAN-GP	WGAN-GP-CT
α_g	0.0002	0.0001	0.0002	0.00005	0.00001	0.0002
α_d	0.0002	0.0001	0.0002	0.00005	0.00001	0.0002
β_1	0.5	0.5	0.5	x	0.5	0.5
β_2	0.999	0.999	0.999	x	0.999	0.999
n_{disc_iters}	1:1	1:1	1:1	5:1	5:1	5:1
λ_1	x	10	10	x	10	10
λ_2	x	x	2	x	2	2
M	x	x	0.2	x	x	0.2

Table 1: Hyperparameters for models not using the Two Times Update Rule

	GAN	GAN-GP	GAN-GP-CT	WGAN	WGAN-GP	WGAN-GP-CT
α_g	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
α_d	0.0004	0.0002	0.0002	0.0002	0.0002	0.0002
β_1	0.5	0.5	0.5	x	0.5	0.5
β_2	0.999	0.999	0.999	x	0.999	0.999
n_{disc_iters}	1:1	1:1	1:1	2:1	2:1	2:1
λ_1	x	10	10	x	10	10
λ_2	x	x	2	x	x	2
M	x	x	0.2	x	x	0.2

Table 2: Hyperparameters for models using the *Two Times Update Rule*

5. Results

In this section, we will analyze our results, first in terms of learning dynamics, and then in terms of driving data generation. By learning dynamics,

we refer to learning stability like learning curve magnitudes, and we consider a model convergence when a state of equilibrium is reached i.e. when both learned curves stabilize in zones of fixed magnitudes.

5.1. Stability and equilibrium

We need to qualitatively analyze the impact of each type of normalization on learning. Our main objectives focus on which normalization methods improve stability and convergence, and whether one of them exceeds the use of regularizations. We can see that the learning dynamics of GANs, like WGANs, are far from stable in all cases (Figures 5, 4, 6, 7). By applying only Batch Normalization (BN), stability is far from being achieved with oscillations the amplitudes of which are extremely high. This is also the case when applying layer normalization (LN). The interest of this method lies in the formulation of the gradient penalty which, in order to preserve the properties on which it is based, prefers its use in the discriminator/critic to batch normalization. GANs do not benefit from easier convergence and the equilibrium state is then less distinct, this phenomenon being observed just as much for WGANs. The layer normalization even seems to increase the oscillations of the generator and the discriminator, compared to the batch normalization. However, this last point must be qualified because the respective oscillations follow generally the same order of magnitude. Using LN either on GANs and WGANs is not theoretically expected to bring about major changes, since the WGAN's formulation, although different from the GAN's one, is not subject to any condition on global normalization. In other words, for the unregularized GANs and WGANs, the batch normalization as well as the layer normalization should not be considered as a source of

learning stability, even if an equilibrium seems to be reached (For WGAN, we have a divergence at the beginning that reaches a balance with strong spaced oscillations).

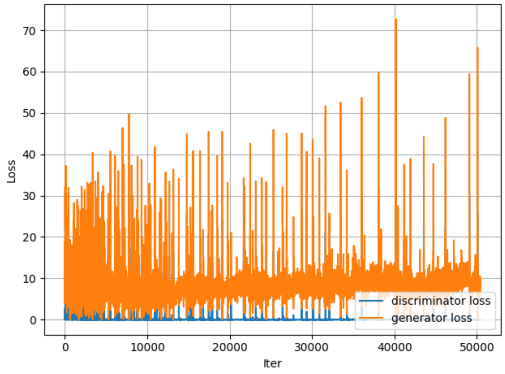

Figure 4: GAN with BN in D and G, TTUR

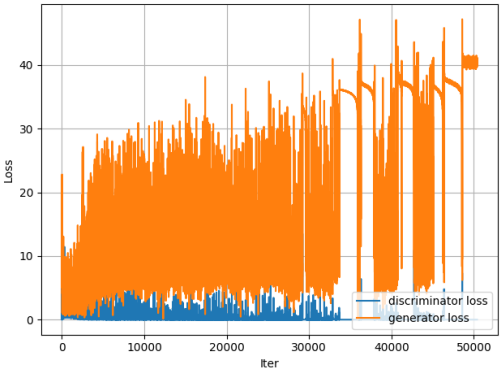

Figure 5: GAN with LN in D and G, TTUR

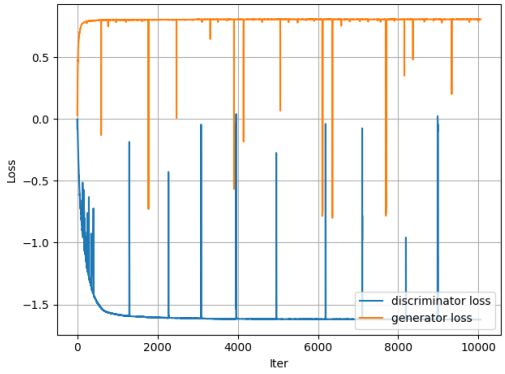

Figure 6: WGAN with BN in D and G

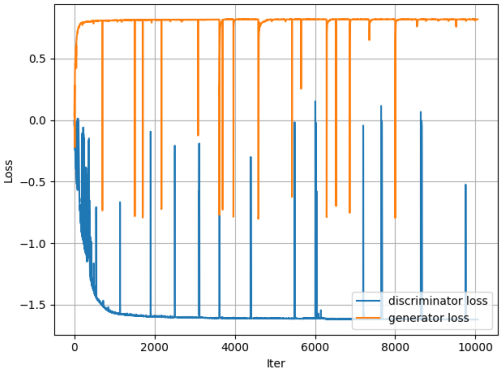

Figure 7: WGAN with LN in D and G

By applying a gradient penalty (GP), learning becomes more stable. If we look at the two previous normalizations, the learning of GANGP is not optimized by a BN. On the other hand, the LN allows a clearer balance to be achieved (the two curves are "one on top of the other") despite the

oscillations, which is logical given its usefulness in this situation. For the WGANGP, the results do not support this assumption, since the curves for the BN and the LN are almost identical. The hypothesis would be that convergence improvements come only from the GP and that LN produces a better generation since it guarantees the validity of the interpolation performed by GP. It should be noted that for LN and BN, the WGAN losses reach very high values. However, the loss of the latter is supposed to be correlated to the quality of the elements generated [6]: this would mean that the model should produce incorrect elements with such behavior. We will see that this conjecture is not proven in our case. Thus, BN and LN do not have a significant impact, and it is worth noting that the GP provides an effective first response to the issue of stability and convergence, especially for the WGANGP. If we apply the consistency term (CT) in addition to the GP, to both GANS and WGANs, we observe that the LN produces the same results as for the GP alone. Knowing that by cross-checking the learning results of a GP with BN, the consistency term is more helpful in achieving a state of equilibrium. Stability appears to be the same as for GP and GPCT models. In this sense, the CT does not seem to bring more than GP to the point of learning itself. GPs and CTs therefore have an impact on the convergence process and stability, but their real appeal would probably lie in the quality of the elements generated.

The most interesting results are achieved by the use of spectral normalization (SN) (Figures 8, 9, 10, 11). In terms of stability and convergence, the latter greatly stabilizes the entire process and makes it possible to achieve a clear equilibrium, regardless of the model chosen and regardless of the

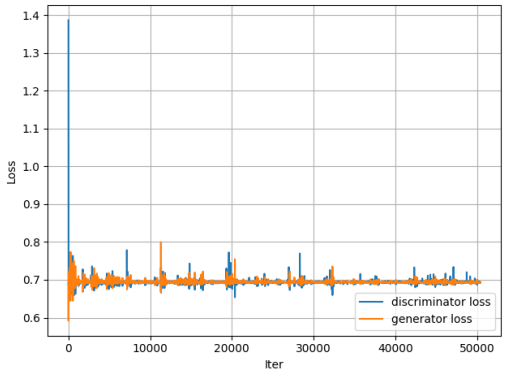

Figure 8: GAN with SN in D and G

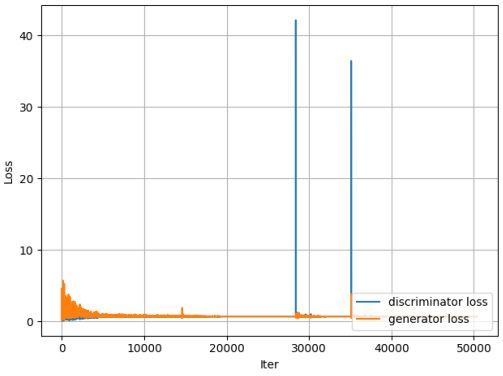

Figure 9: GANGP with SN in D and G

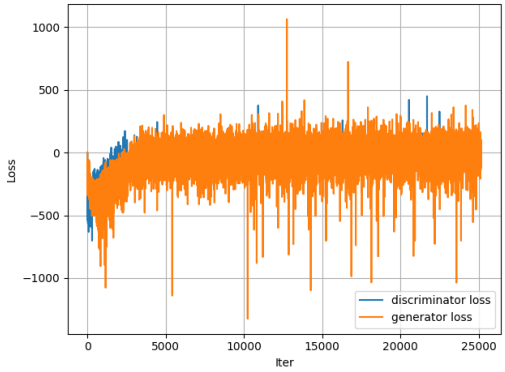

Figure 10: WGAN with SN in D, and TTUR

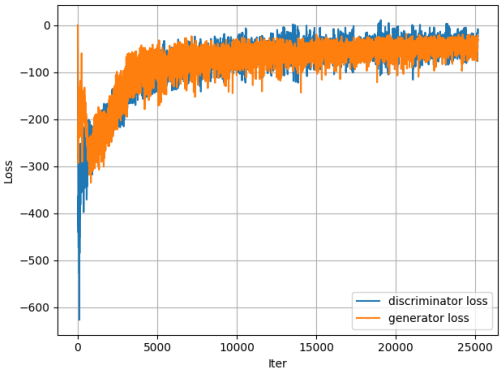

Figure 11: WGANGPCT with SN and dropout in D, and TTUR

placement of the normalization (in D and/or in G, followed by a batch normalization in the generator,...). It should also be noted that coupled with a WGANGP with dropout, it provides exceptional stability. Let us note two other interesting results, the addition of dropout almost always seems to reduce oscillations of the curves, thus improving stability. The TTUR proves what it was introduced for, by facilitating convergence and, above all, by accelerating it. In conclusion, it is certain that spectral normalization

exceeds both regularizations, and this is even truer when combined with the latter. But in addition to the powerful aspect of SN, it would be interesting to observe GP and CT as terms that influence generation quality more than stability and convergence, again given our data. It is of course necessary to look at the generated series and the geometry score to be able to really determine if spectral normalization brings a real plus when it is alone, and if there is a link between adversarial learning dynamics and generated data in the context of scenarios generation.

5.2. Generation quality

The BN and the LN do not bring anything significant in terms of convergence and stability. We observe empirically the same phenomenon regarding the quality of generated data. Indeed, the series derived from GAN models that use these normalizations do not succeed in correctly capturing the entire data distribution and some series are therefore closer to hazardous oscillations than those of interest (Figures 12, 13, 14, 15). It should be noted, however, that the series generated by GANs using the LN or the BN are sometimes realistic. In a way, specific measures could be considered as simpler univariate series, leading to a much better understanding of them by GANs; more complex series are then too difficult to capture. The same idea can be found for the WGANs, which are supposed to be better: not all series are correctly captured, but some are extremely well captured. One might think that this is due to the relative simplicity of these series, when compared to the others forming the scenarios.When applying GP, the results differ. For GANs, the series correspond to sine waves in most cases, indicating that the model failed to capture the data distribution (This behavior is found for any type

of normalization in GANGP). For WGANs, the series are immediately improved (Figures 16, 17, 18,19): this is the logical result of this regularization since the exploration of the two data distributions, or at least areas close to them, is carried out and promotes the Lipschitz continuity necessary for the proper functioning of WGANs. If we apply the CT in addition, we find exactly the same behavior for GANs: the models failed to capture the real data distribution. For WGANs, the generation is further improved, as the exploration of space becomes finer, always in accordance with the formulation of the Critic: indeed, we also observe correlations between the different series. For example, if the curvature of the road is increasing, the speed of the ego vehicle is decreasing. Several noise artifacts are sometimes observed with the gradient penalty, and their presence slowly disappears if we reduce the harshness of the penalty. The same phenomenon occurs when the CT is added, with the same reaction towards changes in its harshness parameter.

However, it should be noted that spectral normalization does not necessarily bring a benefit in terms of generation quality. To be more precise, it should be pointed out that for a GAN without regularization, the SN does not cause any significant change in the quality of generation; similarly, for the GANGP and GANGPCT, SN did not rectify model failures. On the other hand, examples generated from WGAN models that have used SN alone or with regularization tend to be generally more realistic than those generated from models that do not use it. This can be explained simply by the fact that the formulation of spectral normalization favors the WGAN Critic, so learning is more effective, and the addition of regularization to this normalization only improves the exploration of spaces as stated above. The results

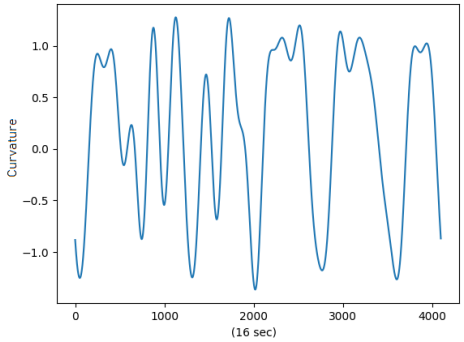

Figure 12: Generated curve of the road's curvature using GAN

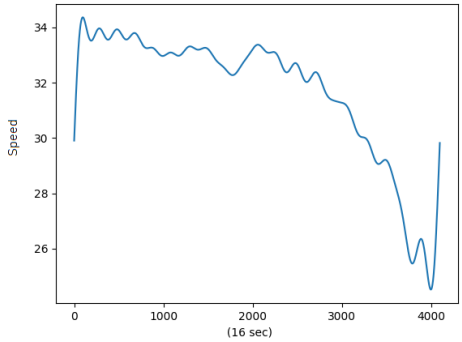

Figure 13: Generated curve of the absolute speed using GAN

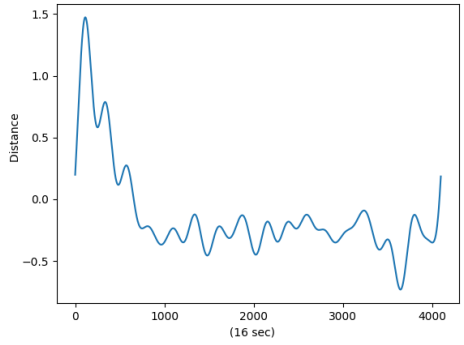

Figure 14: Generated distance from the road center using GAN

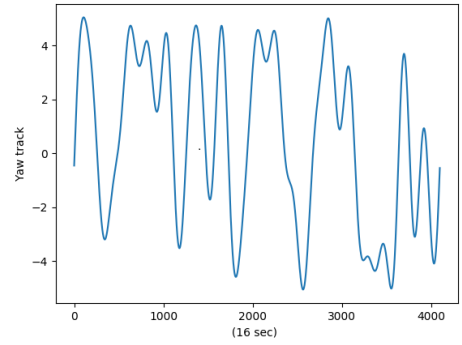

Figure 15: Generated curve of Yaw track using GAN

of the gradient penalty and the consistency term on GANs can be explained by the fact that the latter are calculated from the softmax outputs of the discriminator the inputs of which are normalized between -1 and 1: in theory, the scale of distances would make gradient penalty extremely harsh and would encourage the discriminator to focus only on minimizing the penalty without worrying about its primary task, not letting the consistency term

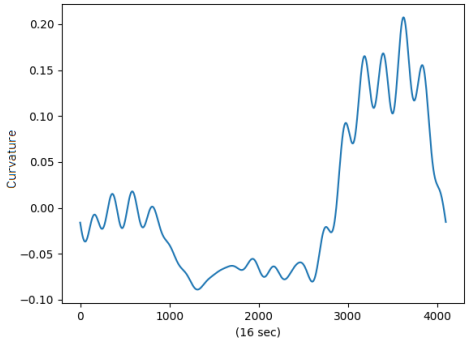

Figure 16: Generated curve of the road's curvature using WGAN

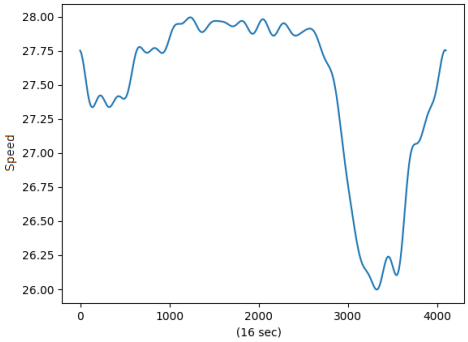

Figure 17: Generated curve of the absolute speed using WGAN

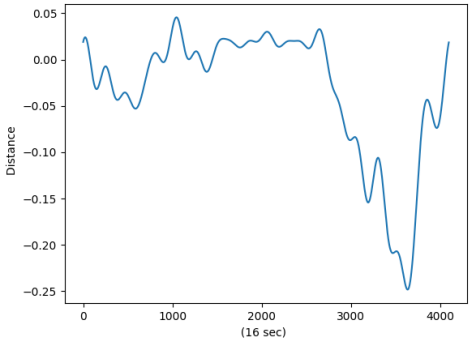

Figure 18: Generated distance from the road center using WGAN

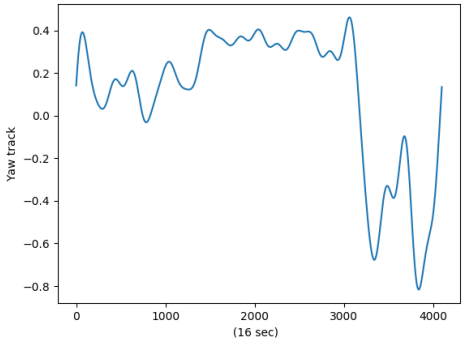

Figure 19: Generated curve of Yaw track using WGAN

bring his added value. In other words, spectral normalization enhances the quality of generation only for WGAN type formulations. In fact, the main observed fact of spectral normalization is its ability to promote the generation of specific scenarios. In the case of traditional GANs, we observe the generation of straight lines, which are very atypical scenarios in the dataset (Figures 20, 21, 22, 23), even if these straight lines could also indicate that

the model overfitted. In the case of WGANs with or without regularization, the same observation applies to atypical scenarios that could be described as *critical*; by this, we mean close to an erratic behavior of the model that would make decisions and then change its mind very quickly on several occasions. Similarly, it also appears to slightly reduce the presence of noise artifacts in some complex series within generated scenarios.

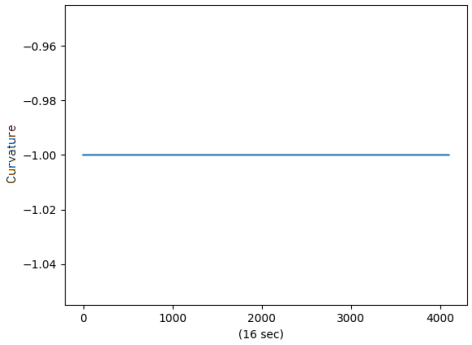

Figure 20: Generated curve of the road's curvature using GAN

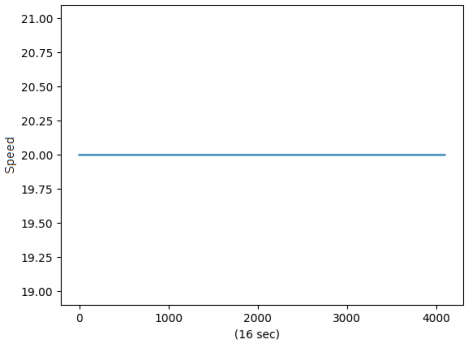

Figure 21: Generated curve of the absolute speed using GAN

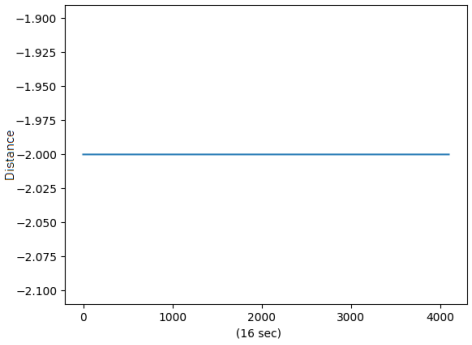

Figure 22: Generated distance from the road center using GAN

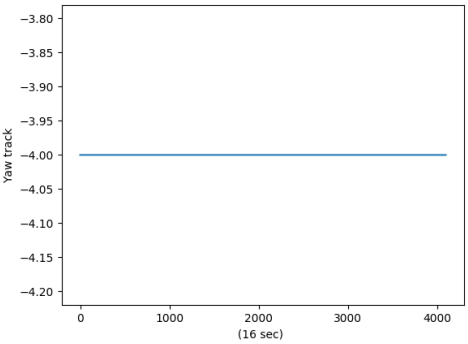

Figure 23: Generated curve of Yaw track using GAN

s

Model	Score	Model	Score	Model	Score
GAN Rd FC 2D	0.306	GANGP RSN FC 2D TTUR	0.353	GANGPCT R FC 2D TTUR	0.342
GAN SN FC 2D	**0.079**	GANGP SN FC 2D	0.340	GANGPCT SN FC 2D TTUR	0.296
GAN LnD FC 2D TTUR	0.14	GANGP Rd FC 2D TTUR	0.297	GANGPCT SnDB FC 2D	0.312
GAN R FC 2D TTUR	0.48	GANGP SN FC 2D TTUR	0.346	GANGPCT SN FC 2D	0.313
GAN SN FC 2D TTUR	0.079	GANGP LnD FC 2D	0.323	GANGPCT Rd FC 2D	0.308
GAN LN FC 2D TTUR	0.077	GANGP SnDB FC 2D TTUR	0.294	GANGPCT LnD FC 2D	0.333
GAN RSnD FC 2D	0.214	GANGP Rd FC 2D	0.344	GANGPCT LN FC 2D TTUR	0.305
GAN R FC 2D	**0.075**	GANGP LN FC 2D TTUR	0.311	GANGPCT LN FC 2D	0.349
GAN SnD FC 2D TTUR	0.095	GANGP RSnD FC 2D TTUR	0.349	GANGPCT RSnD FC 2D$_TTUR$	0.322
GAN SnD FC 2D	0.141	GANGP RSN FC 2D	0.355	GANGPCT RSnD FC 2D	0.304
GAN Rd FC 2D TTUR	0.223	GANGP SnDB FC 2D	0.326	GANGPCT SnD FC 2D TTUR	0.307
GAN LnD FC 2D	0.292	GANGP R FC$_2DTTUR$	0.359	GANGPCT RSN FC 2D TTUR	0.331
GAN RSnD FC 2D TTUR	0.574	GANGP RSnD FC 2D	0.380	GANGPCT SnD FC 2D	0.336
GAN RSN FC 2D TTUR	0.574	GANGP SnD FC 2D	0.327	GANGPCT SnDB FC 2D TTUR	0.345
GAN SnDB FC 2D	0.081	GANGP LN FC 2D	0.252	GANGPCT R FC 2D	0.353
GAN RSN FC 2D	0.57	GANGP SnD FC 2D$_TTUR$	0.296	GANGPCT RSN FC 2D	0.338
GAN LN FC 2D	0.123	GANGP R FC 2D	0.363	GANGPCT Rd FC 2D TTUR	0.279
GAN SnDB FC 2D TTUR	**0.495**	GANGP LnD FC 2D TTUR	0.378	GANGPCT LnD FC 2D TTUR	0.350

Table 3: Geometry scores for the GAN formulation

5.3. Geometry score

The results of the geometry score evaluation conclude this experimental study. We remind you that the lower this score is, the more it reflects the success of GANs learning. Indeed, it corresponds to the "distance" between the approximation of the shape of two simplicial complexes over time (variation of the said parameter ϵ). A low score therefore implies an adversarial generative network that has captured the shape of the manifold on which real data is based and has therefore learned effectively to reproduce the spatial elements of our input data. At the coverage level, this implies that GANs cover the actual data manifold correctly and that the generation coverage level is maximized; indeed, by calculating MRLTs, the shape of the

Model	Score	Model	Score	Model	Score
WGAN RSN FC 2D	0.082	WGANGP SnD FC 2D	0.093	WGANGPCT LnD FC 2D	0.118
WGAN SN FC 2D	0.074	WGANGP SN FC 2D TTUR	0.113	WGANGPCT SnDB FC 2D TTUR	0.116
WGAN RSnD FC 2D TTUR	0.164	WGANGP SnDB FC 2D TTUR	0.075	WGANGPCT LnD FC 2D TTUR	0.077
WGAN LnD FC 2D	0.083	WGANGP R FC 2D	0.092	WGANGPCT SN FC 2D TTUR	0.142
WGAN LnD FC 2D TTUR	0.071	WGANGP RSnD FC 2D	0.079	WGANGPCT LN FC 2D	0.081
WGAN SnDB FC 2D	0.064	WGANGP LnD FC 2D	0.09	WGANGPCT SnDB FC 2D	0.082
WGAN Rd FC 2D	0.061	WGANGP Rd FC 2D TTUR	0.068	WGANGPCT R FC 2D	0.098
WGAN R FC 2D TTUR	0.076	WGANGP R FC 2D TTUR	0.093	WGANGPCT Rd FC 2D	0.065
WGAN RSnD FC 2D	0.144	WGANGP RSnD FC 2D TTUR	0.075	WGANGPCT Rd FC 2D TTUR	0.074
WGAN SN FC 2D TTUR	0.07	WGANGP LN FC 2D TTUR	0.123	WGANGPCT SnD FC 2D	0.077
WGAN LN FC 2D TTUR	0.085	WGANGP RSN FC 2D	0.069	WGANGPCT SN FC 2D	0.081
WGAN RSN FC 2D TTUR	0.080	WGANGP Rd FC 2D	0.079	WGANGPCT SnD FC 2D TTUR	0.068
WGAN Rd FC 2D TTUR	0.124	WGANGP SnD FC 2D TTUR	0.067	WGANGPCT RSN FC 2D	0.131
WGAN SnD FC 2D TTUR	0.170	WGANGP LnD FC 2D TTUR	0.068	WGANGPCT RSN FC 2D TTUR	0.124
WGAN SnDB FC 2D TTUR	0.096	WGANGP LN FC 2D	0.106	WGANGPCT RSnD FC 2D	0.121
WGAN R FC 2D	0.065	WGANGP SN FC 2D	0.109	WGANGPCT R FC 2D TTUR	0.065
WGAN LN FC 2D	0.071	WGANGP SnDB FC 2D	0.067	WGANGPCT RSnD FC 2D TTUR	0.071
WGAN SnD FC 2D	0.102	WGANGP RSN FC 2D TTUR	0.067	WGANGPCT LN FC 2D TTUR	0.071

Table 4: Geometry scores for the WGAN formulation

manifold of generated data approaches the actual data one (number of holes in particular). It is interesting here to start by studying results of spectral normalization. The latter gives good geometry scores, but not the best in all types of formulations: in this sense, the SN improves the stability and convergence of the model, as well as the quality of the generation because it values 1-Lipschitz continuity, and this valorization is found in the geometry score of all models that use it. In such a case, the latter is generally low, except for the GANGP and the GANGPCT. The qualitative study of the scenarios generated by the latter revealed the failure of the learning: the geometry score is therefore necessarily much higher for all the models concerned (Table 3) (values higher than 0.3, compared to values lower than 0.1 for many other models). Empirically, it is fairly easy to conclude that, on

average, models using the Wasserstein formulation are more able to address the coverage problem. Indeed, the Earth Mover distance seems more suitable (it is used in topological data analysis). The corresponding geometry scores are all low (Table 4) and the use of GP and CT is really important: by exploring manifolds more effectively, these regularizations ensure that the number of areas of space that were used to learn distributions is maximized, and that the results of the persistence diagrams coincide over time. Thus, all WGANGPs and WGANGPCTs achieve a similar level of coverage, without focusing on any particular normalization. The latter once again addresses the problem of convergence and stability. We can therefore say that the level of coverage becomes better thanks to regularization.

6. Conclusions and future works

The generation of autonomous driving scenarios via generative adversarial networks is possible and the elements produced are realistic with regard to simulation data, attesting to a true interest in the method. The gradient penalty and the consistency term are of great interest when trying to achieve a better level of manifold coverage. In the case of WGANs, they guarantee slightly more realistic series by exploring manifolds and the level of coverage becomes better on the whole, despite certain risks with regard to noise. A WGANGP or a WGANGPCT will give a satisfactory level of coverage, the SN guaranteeing a certain stability if it is used again. The geometry score made it possible to formalize measurements of the data coverage achieved by generation to compare the different GANs.

Create a score that would measure the level of coverage beyond the manifold, potentially on its external borders for unknown scenarios, would better address the issue of evaluating levels of coverage achieved by the generation. Finally, the noise removal process was designed to objectively study the generation of each model, but it would be beneficial to deepen it, in order to build a more rigorous signal processing.

7. Acknowledgements

The authors wish to express their thanks for the financial support of the Institute for Technological Research (IRT) Systemx

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[3] M. Arjovsky, L. Bottou, Towards Principled Methods for Training Generative Adversarial Networks, arXiv e-prints (2017).

[4] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, CoRR abs/1606.03498 (2016).

[5] M. Luke, P. Ben, P. David, S. D. Jascha, Unrolled generative adversarial networks, CoRR abs/1611.02163 (2016).

[6] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN, arXiv e-prints (2017).

[7] C. Villani, Optimal Transport: Old and New, Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg, ????

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. C. Courville, Improved training of wasserstein gans, in: Advances in neural information processing systems, 2017, pp. 5767–5777.

[9] X. Wei, B. Gong, Z. Liu, W. Lu, L. Wang, Improving the improved training of wasserstein gans: A consistency term and its dual effect, arXiv preprint arXiv:1803.01541 (2018).

[10] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, I. Goodfellow, Many paths to equilibrium: Gans do not need to decrease a divergence at every step, arXiv preprint arXiv:1710.08446 (2017).

[11] Y. Saatci, A. G. Wilson, Bayesian gan, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 3622–3631.

[12] L. Chongxuan, T. Xu, J. Zhu, B. Zhang, Triple generative adversarial nets, in: Advances in neural information processing systems, 2017, pp. 4088–4098.

[13] C. Wang, C. Xu, X. Yao, D. Tao, Evolutionary generative adversarial networks, CoRR abs/1803.00657 (2018).

[14] A. B. L. Larsen, S. K. Sønderby, O. Winther, Autoencoding beyond pixels using a learned similarity metric, CoRR abs/1512.09300 (2015).

[15] M. Lucic, K. Kurach, M. Michalski, S. Gelly, O. Bousquet, Are gans created equal? a large-scale study, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31, Curran Associates, Inc., 2018, pp. 700–709.

[16] K. Kurach, M. Lucic, X. Zhai, M. Michalski, S. Gelly, The GAN landscape: Losses, architectures, regularization, and normalization, CoRR abs/1807.04720 (2018).

[17] K. Roth, A. Lucchi, S. Nowozin, T. Hofmann, Stabilizing training of generative adversarial networks through regularization, in: Advances in neural information processing systems, 2017, pp. 2018–2028.

[18] N. Kodali, J. D. Abernethy, J. Hays, Z. Kira, How to train your DRAGAN, CoRR abs/1705.07215 (2017).

[19] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, CoRR abs/1802.05957 (2018). URL: `http://arxiv.org/abs/1802.05957`.

[20] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).

[21] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167 (2015).

[22] L. Mescheder, A. Geiger, S. Nowozin, Which training methods for gans do actually converge?, arXiv preprint arXiv:1801.04406 (2018).

[23] V. Nagarajan, J. Z. Kolter, Gradient descent gan optimization is locally stable, in: Advances in Neural Information Processing Systems, 2017, pp. 5585–5595.

[24] H. Zhang, I. Goodfellow, D. Metaxas, A. Odena, Self-attention generative adversarial networks, arXiv preprint arXiv:1805.08318 (2018).

[25] A. Brock, J. Donahue, K. Simonyan, Large scale gan training for high fidelity natural image synthesis, arXiv preprint arXiv:1809.11096 (2018).

[26] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, NIPS (2014).

[27] L. Yu, W. Zhang, J. Wang, Y. Yu, Seqgan: Sequence generative adversarial nets with policy gradient, CoRR abs/1609.05473 (2016).

[28] M. J. Kusner, J. M. Hernández-Lobato, Gans for sequences of discrete elements with the gumbel-softmax distribution, CoRR (2016).

[29] D. Donahue, A. Rumshisky, Adversarial text generation without reinforcement learning, CoRR abs/1810.06640 (2018).

[30] C. Esteban, S. L. Hyland, G. Rätsch, Real-valued (medical) time series generation with recurrent conditional gans, arXiv preprint arXiv:1706.02633 (2017).

[31] O. Mogren, C-RNN-GAN: continuous recurrent neural networks with adversarial training, CoRR abs/1611.09904 (2016).

[32] O. Press, A. Bar, B. Bogin, J. Berant, L. Wolf, Language generation with recurrent generative adversarial networks without pre-training, CoRR abs/1706.01399 (2017).

[33] Y. Luo, X. Cai, Y. Zhang, J. Xu, et al., Multivariate time series imputation with generative adversarial networks, in: Advances in Neural Information Processing Systems, 2018, pp. 1596–1607.

[34] S. Takahashi, Y. Chen, K. Tanaka-Ishii, Modeling financial time-series with generative adversarial networks, Physica A: Statistical Mechanics and its Applications 527 (2019) 121261.

[35] A. S. Koshiyama, N. Firoozye, P. C. Treleaven, Generative adversarial networks for financial trading strategies fine-tuning and combination, CoRR abs/1901.01751 (2019).

[36] M. Wiese, R. Knobloch, R. Korn, P. Kretschmer, Quant GANs: Deep Generation of Financial Time Series, arXiv e-prints (2019).

[37] D. Lopez-Paz, M. Oquab, Revisiting classifier two-sample tests, arXiv preprint arXiv:1610.06545 (2016).

[38] K. Shmelkov, C. Schmid, K. Alahari, How good is my gan?, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 213–229.

[39] R. Fortet, E. Mourier, Convergence de la répartition empirique vers la répartition théorique, in: Annales scientifiques de l'École Normale Supérieure, volume 70, 1953, pp. 267–285.

[40] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, Journal of Machine Learning Research 13 (2012) 723–773.

[41] D. J. Im, H. Ma, G. Taylor, K. Branson, Quantitatively evaluating gans with divergences proposed for training, arXiv preprint arXiv:1803.01045 (2018).

[42] A. Borji, Pros and cons of gan evaluation measures, Computer Vision and Image Understanding 179 (2019) 41–65.

[43] L. Theis, A. van den Oord, M. Bethge, A note on the evaluation of generative models, arXiv e-prints (2015) arXiv:1511.01844. arXiv:1511.01844.

[44] R. Ghrist, Barcodes: the persistent topology of data, Bulletin of the American Mathematical Society 45 (2008) 61–75.

[45] V. Khrulkov, I. V. Oseledets, Geometry score: A method for comparing generative adversarial networks, CoRR abs/1802.02664 (2018).

[46] Z. Wang, T. Oates, Encoding time series as images for visual inspection and classification using tiled convolutional neural networks, in: Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[47] E. Brophy, Z. Wang, T. E. Ward, Quick and easy time series generation with established image-based gans, CoRR abs/1902.05624 (2019).

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: