Recurrent Conditional Generative Adversarial Networks for Autonomous Driving Sensor Modelling

Henrik Arnelid* [1], Edvin Listo Zec* [1], Nasser Mohammadiha[3]

***Abstract*— Simulation of the real world is a widely researched topic in various fields. The automotive industry in particular is very dependent on real world simulations, since these simulations are needed in order to prove the safety of advance driver assistance systems (ADAS) and autonomous driving (AD). In this paper we propose a deep learning based model for simulating the outputs from production sensors used in autonomous vehicles. We introduce an improved Recurrent Conditional Generative Adversarial Network (RC-GAN) consisting of Recurrent Neural Networks (RNNs) that use Long Short-Term Memory (LSTM) in both the generator and the discriminator networks in order to generate production sensor errors that exhibit long-term temporal correlations. The network is trained in a sequence-to-sequence fashion where we condition the output from the model on sequences describing the surrounding environment. This enables the model to capture spatial and temporal dependencies, and the model is used to generate synthetic time series describing the errors in a production sensor which can be used for more realistic simulations. The model is trained on a data set collected from real roads with various traffic settings, and yields significantly better results as compared to previous works.**

I. Introduction

A lot of progress is continuously being made in the race to autonomy in the automotive industry. Many advance driver assistance system (ADAS) technologies, such as lane keeping assist, collision warning and emergency braking are already implemented in vehicles today and are foreseen to reduce the risk of accidents on roads. In order to make a safe decision, the vehicles are installed with a lot of different sensors such as cameras, radars and lidars. The sensor data is often fused in order for the vehicle to have as much information as possible of its surrounding 360 degree environment.

In order to guarantee the safety of the autonomous vehicles with a certain confidence, billions of miles have to be driven in order to provide a good statistic for a low fatality rate [1]. This is a very time consuming and expensive task which cannot be done in reality. Companies developing software for ADAS and autonomous driving (AD) thus resort to virtual verification, where models of the environment and sensors are made to match reality. Modelling a sensor can be done in many different ways on different levels. For example, it is possible to model the raw detections of a sensor. In this paper we focus on object level data, which is the output of sensor fusion. Modelling sensor characteristics such as sensor errors is a challenging problem because it requires models that can capture stochastic behaviours of sensors. Furthermore, it is important to understand how the errors correlate with different traffic scenarios and settings.

Generative models are a fundamental part in a lot of different machine learning algorithms and the attention to generative models is increasing, a lot due to their capability of modelling underlying statistical structures of high dimensional signals. Especially in computer vision, Generative Adversarial Networks (GANs) [2] have had great success recently generating realistic high-quality images. Inspired from this, in this paper we develop a deep learning based sensor model capable of modelling the continuous sequential data describing errors in sensor outputs using adversarial networks. Two main properties of sensor errors are temporal correlations and dependencies on other parameters [3]. To capture these properties, we use a continuous-valued conditional GAN where we condition the output of the model with input features as well as introducing Recurrent Neural Networks in both the generator and the discriminator.

The main contribution in this paper is an improved version of the recurrent and conditional Generative Adversarial Network [4] which we implement in order to model sensors used in autonomous vehicles. In this paper, the model is applied to an ADAS and AD sensor, however it can be used for any type of real-valued signal.

II. Related Work

The work in this paper is related to previous works on sensor modelling and also to Generative Adversarial Networks and Recurrent Neural Networks for sequence generation and prediction. Since the introduction of GANs, they have shown good results in generating realistic samples in many different applications [5], [6], [7], where tasks involving images are predominant. However, GANs have not been explored as extensively to generate synthetic time series. GANs have previously been used for sequential data generation, but these typically focus on discrete outputs such as in language processing [8]. The amount of research for generating continuous real-valued time series using GANs is limited as compared to image generation. In [4], [9], [10] the authors use RNN based GANs in order to model continuous time series. In [11] they modify already existing image generation methods to operate on audio waveforms, which is a different approach than using GANs for modelling multivariate time series.

*These authors contributed equally to this paper.

[1]Henrik Arnelid is with Zenuity AB, Gothenburg, Sweden henrik.arnelid@zenuity.com

[1]Edvin Listo Zec is with RISE AI. Edvin was with Zenuity AB when the main work was performed, and was continued at RISE. edvin.listo.zec@ri.se

[3]Nasser Mohammadiha is with Zenuity AB, Gothenburg, Sweden nasser.mohammadiha@zenuity.com

An Autoregressive Input-Output Hidden Markov Model (AIO-HMM) for generation of real-valued time series describing sensor errors has recently been proposed in [3]. Given input features describing the environment, the authors managed to generate time series of sensor errors similar to that exhibited by a production sensor. Given the recent success of GANs in the image domain, we extend upon the idea of the AIO-HMM and show that it is possible to improve the realism of the generated time series with a improved version of the RC-GAN as described in [4].

A. Problem Overview and Data Set

The output from sensors used in ADAS and AD considered in this paper is in the form of dynamic state vectors over time, describing variables such as object position, velocity and acceleration relative to the host vehicle collecting the sensor data. These outputs from production sensors inherently exhibit noise and inaccuracies. The main contribution in this paper is to create a model for generating synthetic but realistic production sensor outputs. Particularly, we focus on modelling the time series describing the longitudinal and lateral position and velocity errors from the sensors. A trained model can then be used in Computer Aided Engineering (CAE) tools for virtual testing.

In the same fashion as in [3], we use a radar and camera fusion based production sensor setup in our experiments. Moreover, the ego vehicle is also equipped with a Velodyne lidar HDL-64E which is used as a reference system. The lidar data is processed using object classification and tracking algorithms and the output is in the form of object lists with estimated properties like position and speed. We define the production sensor error as the difference between the production sensor and the reference sensor outputs for every detected object over time. Let $\boldsymbol{x}_*$ be multi-dimensional sequences describing sensor outputs, such as position and speed for target objects. For a given sequence $\boldsymbol{x}$ that we want to model, the error over time t is defined as

$$\boldsymbol{\varepsilon}(t) = \boldsymbol{x}_{\text{sensor}}(t) - \boldsymbol{x}_{\text{reference}}(t). \tag{1}$$

In order to calculate the error we need to associate each production sensor object with a corresponding reference system object. We do this by using an offline matching algorithm [12], where tracked objects are represented with a dynamic state vector with the objects position, speed, acceleration, width etc. The output from the matching algorithm is a matrix consisting of object properties. These properties are in the form of time series for each uniquely tracked object, both from the reference system and the production sensor. The data set that we use consists of sensor outputs collected from four days of driving on European highways and trunk roads, spanning over 12,000 multivariate time series, describing variables such as position, speed, heading etc. The validation data set contains around 2,000 multivariate time series. The average length of all time series is 197 frames with a minimum of 50 and maximum of 828 frames. In order to enrich the scenario space of the data set, different type of vehicle scenarios such as cut-ins, overtakes and trailings are present in the data set.

III. Proposed Model

A. Recurrent Neural Networks

The vanilla RNN is a non-linear mapping f which takes a vector sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ as input and outputs a high-level representation sequence $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T)$ [13]. The equations describing the mapping are

$$\begin{aligned} \boldsymbol{h}_t &= f(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{H}\boldsymbol{h}_{t-1} + \boldsymbol{b}), \\ \boldsymbol{p}_t &= \texttt{softmax}(\boldsymbol{W}_p\boldsymbol{h}_t + \boldsymbol{b}_p), \end{aligned} \tag{2}$$

where $\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{b}, \boldsymbol{W}_p, \boldsymbol{b}_p$ are learnable parameters of the network, and $\boldsymbol{p}_t$ is the softmax probability of having seen the observations up to $\boldsymbol{x}_t$.

B. Long Short-Term Memory Units

A drawback of using vanilla RNNs that are trained with gradient descent methods is the vanishing gradient problem where the gradients corresponding to past observations become vanishingly small and weights do not get updated properly. One solution to this problem is the Long Short-Term Memory network which incorporates a memory cell $\boldsymbol{c}$ together with an input gate $\boldsymbol{i}$, an output gate $\boldsymbol{o}$ and a forget gate $\boldsymbol{f}$ [14]. The memory cell enables the network to remember its state over time, and by doing so it is possible for the full network to capture long-term temporal dependencies present in the training data. Let $\odot$ be the Hadamard product, σ an activation function and $\tanh(\cdot)$ be applied element-wise. The computational flow of an LSTM is then as follows:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{V}_i\boldsymbol{c}_{t-1} + \boldsymbol{b}_i) \tag{3}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{V}_f\boldsymbol{c}_{t-1} + \boldsymbol{b}_f) \tag{4}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{5}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{V}_o\boldsymbol{c}_t + \boldsymbol{b}_o) \tag{6}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{7}$$

where $\boldsymbol{W}_*, \boldsymbol{V}_*, \boldsymbol{U}_*$ and $\boldsymbol{b}_*$ are learnable parameters. In Equations (3) and (4) the input and the forget gate are computed, which are then used to update the memory cell $\boldsymbol{c}$ in Equation (5). Then the output gate is computed in Equation (6), and lastly the final output $\boldsymbol{h}_t$ is computed in Equation (7).

C. Generative Adversarial Networks

A Generative Adversarial Network is a generative neural network that aims to generate samples given a distribution $p_{\text{data}}(\boldsymbol{x})$ of the training data. In the GAN architecture there are two different neural networks trained simultaneously, a generator $G(\boldsymbol{z}; \boldsymbol{\theta}_g)$ and a discriminator $D(\boldsymbol{x}; \boldsymbol{\theta}_d)$, which have conflicting objectives. The generator learns a distribution p_g over the data $\boldsymbol{x}$, whereas the goal of the discriminator is to discriminate between the synthetic data $G(\boldsymbol{z})$ generated by

G and the real data $\boldsymbol{x}$. In practice, this is a minimax game problem described with the value function $V(D, G)$ as

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{8}$$

Here, we have defined a prior distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$ over the input noise variables. It has been shown that the minimax game has a global optimum for $p_g = p_{\text{data}}$ [2].

D. Recurrent Conditional Generative Adversarial Networks

The model implemented in this paper is based on the networks described in [4]. The first difference from the original GAN that the authors propose in their paper is that both the generator and discriminator are replaced by RNNs with LSTM units. The second change is that the output from both the generator and the discriminator is conditioned on an input vector $\boldsymbol{y}$, which makes it possible for the GAN to learn the conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{y})$ as described in [2]. The value function is then expressed as

$$V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{y})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})|\boldsymbol{y}))]. \tag{9}$$

Figure 1 depicts the network architecture. In the same manner as with the original GAN, the generator takes a latent vector $\boldsymbol{z}$ sampled from a known distribution as input. Furthermore, the discriminator takes either a real or synthetic sequence as input where the network outputs a softmax probability at each time step classifying if the sample is real or synthetic. All predictions for each time step are then used to calculate the loss function. Finally, both G and D are conditioned on a multi-dimensional real-valued sequence $\boldsymbol{y}$. In our case, we use real-valued sequences for the output, input and the latent space denoted by

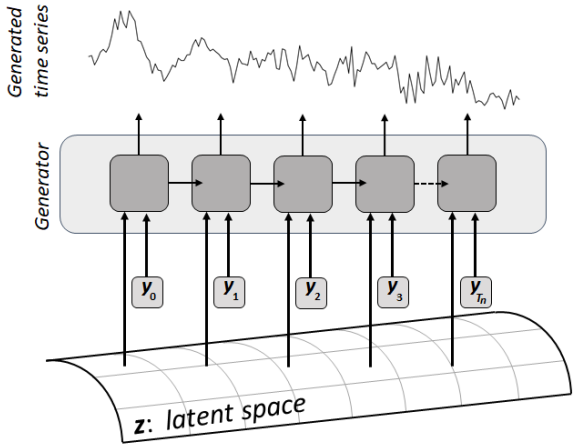
$$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\} \tag{10}$$

$$\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T\} \tag{11}$$

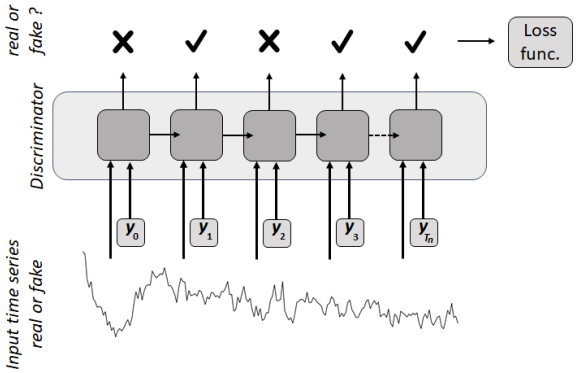
$$\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_T\}, \tag{12}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{T_t \times k}$, $\boldsymbol{y}_t \in \mathbb{R}^{T_t \times \ell}$, and $\boldsymbol{z}_t \in \mathbb{R}^{T_t \times m}$ where T_t is the length of the series for $t = 1, 2, \ldots, T$ and k, ℓ and m are the feature dimensions.

One main change in our model as compared to [4] is that we isolate the latent noise to its own RNN. In our case, the generator consists of two sets of RNNs. One RNN that the latent vector $\boldsymbol{z}_t$ is fed to, and one that the conditional input is fed to.



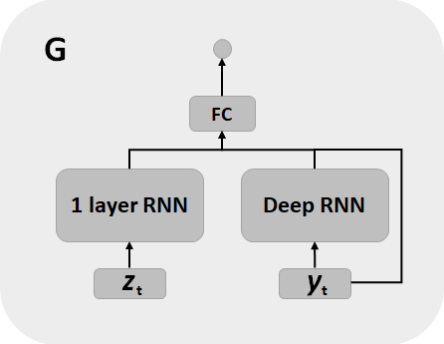
(a) Generator network (G). It takes input from a latent space as well as condition data $\boldsymbol{y}_t$ at each time frame.



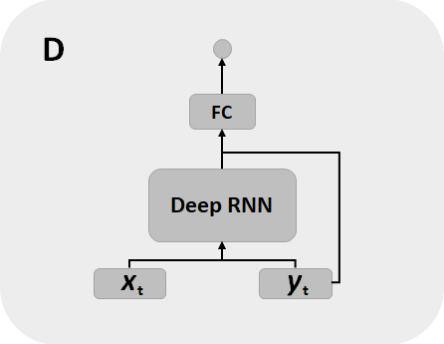
(b) Discriminator network (D). It takes either a real or synthetic time series together with the condition data $\boldsymbol{y}_t$ as input at each time frame.

Fig. 1: The architecture of the generator (top) and discriminator (bottom). Figure is a modified version as seen in [4].

Another change in our model is that we add a skip-connection in both the generator and the discriminator. This allows the networks to predict from both memory and information of the current time-step. The output from both RNNs within and the skip-connected conditional input in G are concatenated and fed into a fully connected layer and the final output is a linear activation. By isolating $\boldsymbol{z}_t$ it is possible to control the amount of noise that is applied to the output in much greater detail through varying the distribution that $\boldsymbol{z}_t$ is sampled from, the network size and the size of $\boldsymbol{z}_t$. The discriminator has a simple structure where the data point(s) from either a real or synthetic time series $\boldsymbol{x}_t$ is being concatenated with the corresponding condition data $\boldsymbol{y}_t$ at each time frame and is fed to a deep RNN. By adding the skip connection for $\boldsymbol{y}_t$ to the fully connected layer for both networks, a more stable and faster learning was obtained during the training. The final model used in this paper consisted of a 2 layered RNN-LSTM in both the generator and the discriminator. The full model architecture proposed in this paper is visualised in Figure 2.



(a) The generator takes a sample $\boldsymbol{z}_t$ from the latent space which is passed to a single layered RNN, while the condition data $\boldsymbol{y}_t$ is fed to a multi-layered RNN. The output from both RNNs and the skip-connected condition data are all concatenated and fed into a fully connected layer and the final output is a linear activation.



(b) The discriminator takes a sample $\boldsymbol{x}_t$ from a time series, real or fake, together with the condition data $\boldsymbol{y}_t$. These two inputs are concatenated and passed to a multi-layered RNN whose output goes into a fully connected layer together with the skip-connected condition data. Finally there is one output neuron to classify the specific sample.

Fig. 2: Internal structure of the generator (top) and discriminator (bottom).

IV. Model evaluation

Evaluation of generative models in general and GANs in particular is an open research question that is far from solved [15]. Especially, a lot of work is done in evaluating the quality of generated images from GANs. Current evaluation still relies on human validation to judge if the generated sample is good enough. However, evaluating and quantifying the quality of generated continuous sequential data is even harder for a human to do. Further, in our case we seldom have more than one realisation from an underlying unknown stochastic sensor model. Using only one sample makes it difficult to draw any reasonable conclusions of the model and its quality.

We evaluate our model by using a validation set including around 2,000 sequences, each corresponding to unique traffic scenarios. In order to evaluate if different parts of the signals, such as large errors, are represented with the right density in the generated samples, we use the Jensen-Shannon distance (JSd) [16]. Further, we also use the root mean squared error (RMSE) to assess the quality of generated sequences by the model.

JSd is a smoother version of the Kullback-Leibler (KL) divergence, which for two distributions P and Q is defined as

$$K(P\|Q) = \sum_x \log\left(\frac{p(x)}{q(x)}\right) p(x). \tag{13}$$

However, the KL divergence is not a true metric since it does not fulfil the triangle inequality and since it is not symmetric. Furthermore, it can yield infinite values when P generates samples that have probability zero for the Q distribution. Thus, we rely on the JSd, which is the square root of the Jensen-Shannon divergence (JSD) and fulfils all metric properties. Let $M = \frac{1}{2}(P + Q)$. The JSD is then defined as

$$J(P\|Q) = \frac{1}{2}K(P\|M) + \frac{1}{2}K(Q\|M). \tag{14}$$

Given that one uses the base 2 logarithm, the JSd will range from 0 to 1, with 0 meaning that P and Q are almost surely identical distributions and 1 meaning that the two distributions are disjoint.

With the RMSE capturing the temporal differences of the time series and the JSd capturing the difference in distributions, we get a solid idea of how well the model performs. In addition to this, we also take into account the JSd between the first difference distributions of the validation set and the generated samples, i.e. the distribution of $x_t - x_{t-1}$, which gives us more understanding of the temporal aspects of the model. In the end we also qualitatively evaluate the models by performing visual inspection of the generated data.

V. Results and Discussion

We randomly initialised and trained several different improved RC-GANs on the same data set, and evaluated each model using the JSd and the RMSE as well as performing visual inspection.

In Figure 3, three different validation sequences together with corresponding generated sequences from the best performing model are visualised. These sequences were chosen to illustrate different behaviour in the data and how the trained model performs in these cases. The validation sequence of the longitudinal position error is plotted with a black line, while the mean of 100 generated sequences from the best RC-GAN is plotted with a blue line. The filled blue area is the 95th percentile of all generated sequences.

All trained models got a substantial initial transient for all generated time series. By performing visual inspection we see that other than the initial transient the generated sequences behave similar to the real sequences. This transient makes the RC-GAN behave poorly with generating first time frames of the generated sequences as compared with the real data, as seen in Figure 3. Furthermore the transient is similar for many different time series which is an undesirable trait since it does not reflect the real data. This issue mainly stems from the LSTM nodes in the network whose internal states have to be built up from consecutive inputs. When the

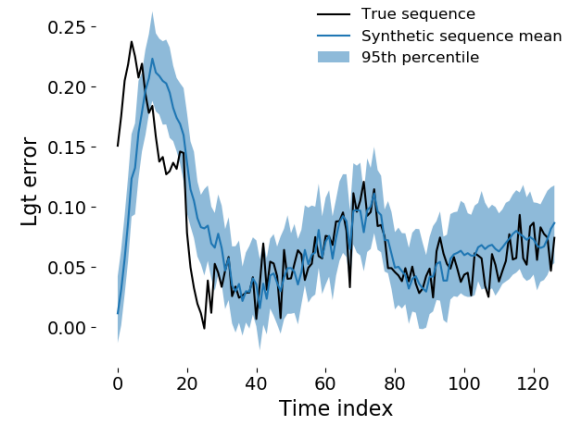
network has been fed enough time frames, they start to be able to grasp the context of the sequences and the output start to look more like the real sequences. Attempts to solve the issue with the initial transient have been made by trying different initialisation schemes, e.g. initialising the LSTMs with noise from either uniform or normal distributions. This does indeed cause the transient to behave differently both for good and for bad. However, the main issue with this approach is that the learning of the model in the GAN setting is slowed down greatly and becomes even more unstable.

We also trained the generator network G using the mean average error (MAE) as loss function instead of the GAN training setting, i.e. without the discriminator. These two models together with the AIO-HMM and the original RC-GAN are compared in Table I. Figure 4 depicts the histogram of the generated data and the real data. In Figure 4a, the generated sensor error distribution is depicted in orange and the distribution of the real sequences is depicted in blue. In Figure 4b, the distributions of the first difference of both the generated sequences and the real data are depicted.

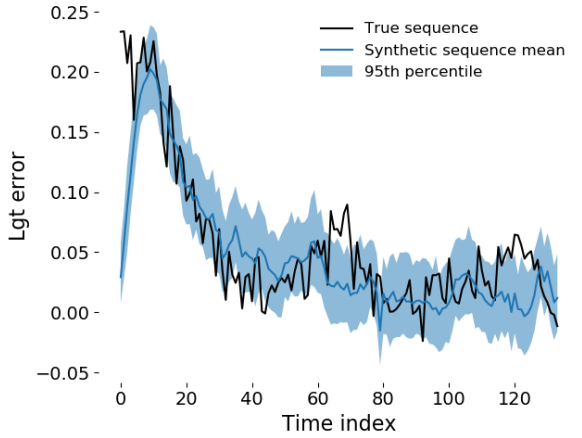
Figure 4 and Table I show that the generated and real data distributions are very similar with a JSd of 0.082 and 0.11 for the total distribution and for the first difference distribution respectively. The results show that the RC-GAN yields a 37% better result as compared to the AIO-HMM for the JSd of the total distribution. For the JSd of the first difference distribution we observe a 27% better performance and a 25% lower RMSE for the RC-GAN as compared to the AIO-HMM. By using consecutive non-linear transformations and internal memory, the improved RC-GAN is able to learn rich representations and model long-term dependencies better than the other models. As compared to the original RC-GAN, we note that we yield a large improvement with respect to JSd and first difference JSd. This is mainly due to the isolation of the noise to its own network and by adding skip connections. With regards to RMSE, we perform a bit worse. This can be explained by that that the original RC-GAN is more unstable in its training, and predicts sequence values close to the sequence mean. The improved RC-GAN is however able to model fluctuations in the sequence.

TABLE I: Table showing metrics showing the results for the AIO-HMM, Generator network G and the RC-GAN.

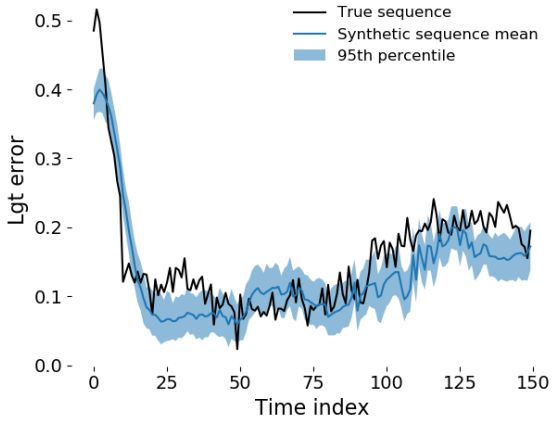
Model	JSd	1^{st} diff JSd	RMSE
AIO-HMM	0.130	0.150	0.670
Original RC-GAN	0.337	0.527	0.448
G (MAE)	0.342	0.550	**0.392**
Our RC-GAN	**0.082**	**0.110**	0.503



(a)

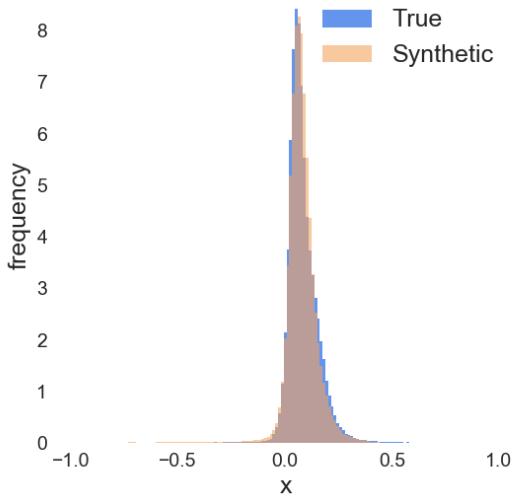


(b)

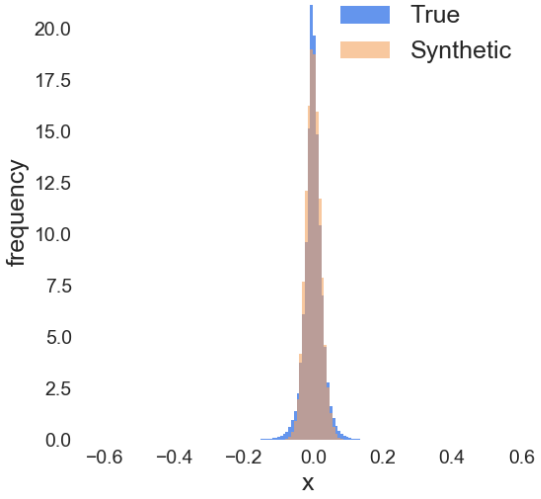


(c)

Fig. 3: Sequences describing the longitudinal position error for three different tracked objects from the validation set (black) together with the mean (blue) and 95th percentile (filled blue) of the 100 generated sequences from the RC-GAN. The vertical axis has been re-scaled.



(a) Histogram of the sensor errors.



(b) Histogram of the first difference of the errors.

Fig. 4: Histogram of the sensor error for all sequences from the validation set (blue) together with the histogram of the generated sensor errors for all sequences (orange) from the RC-GAN. The vertical axis shows a normalised frequency and the horizontal axis has been re-scaled.

VI. Conclusions

In this paper, an improved Recurrent Conditional Generative Adversarial Network (RC-GAN) has been proposed for the purpose of modelling continuous sequences describing sensor outputs that are used in autonomous driving. The RC-GAN is able to handle time series of arbitrary length. Further, by separating the input noise and conditional input to different networks, the model has the ability to tune the noise levels to the specific data distribution that is wished to be learned and generate more stable signals. As it is possible to run the network on arbitrary long sequences it is also possible to train the network on data sets containing sequences of different lengths, which we do in this paper. The proposed model learns the data distribution of time series with long-term temporal dependencies similar to that shown in the real data. In particular, we yield better results than those reported in previous work by a great margin.

Acknowledgement

The authors would like to thank Volvo Car Group for providing the data used for this paper.

References

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

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

[3] E. Listo Zec, N. Mohammadiha, and A. Schliep, "Statistical sensor modelling for autonomous driving using autoregressive input-output hmms," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

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

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[6] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network.," in *CVPR*, vol. 2, p. 4, 2017.

[7] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," in *Advances in Neural Information Processing Systems*, pp. 217–225, 2016.

[8] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient.," in *AAAI*, pp. 2852–2858, 2017.

[9] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

[10] H. Arnelid, "Sensor modelling with recurrent conditional gans," Master's thesis, Chalmers University of Technology, 2018.

[11] C. Donahue, J. McAuley, and M. Puckette, "Synthesizing audio with generative adversarial networks," *CoRR*, vol. abs/1802.04208, 2018.

[12] J. Florbäck, L. Tornberg, and N. Mohammadiha, "Offline object matching and evaluation process for verification of autonomous driving," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pp. 107–112, IEEE, 2016.

[13] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, pp. 1310–1318, 2013.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv preprint arXiv:1511.01844*, 2015.

[16] J. Briët and P. Harremoës, "Properties of classical and quantum jensen-shannon divergence," *Physical review A*, vol. 79, no. 5, p. 052311, 2009.