The GANbler: learning, representing and generating gambling behavior with neural networks

Vadim Kufenko & Steffen Otterbach

UNIVERSITÄT HOHENHEIM

Work in progress / Please do not quote without permission

Teaser

Note: due to data protection the anonymous real data used for visualization has been slightly distorted.

This research has been solely funded by the University of Hohenheim.

Contents

Introduction and Motivation

Background

- Machine and deep learning models require large amounts of data.
- Regarding empirical research the scarcity of problem gambling data implies:
  - unequal class distribution in cross-sectional data;
  - scarcity of time-series sequences;
  - limited knowledge of the distributions.
- Known solutions:
  - Cross-section: 80+ synthetic oversampling methods (Kovács, 2019) including deep learning applications (Pathare et al., 2023);
  - Time-series: Generative Adversarial Networks (GANs) with recurrent and/or convolutional neural networks and different architecture (Brophy et al., 2023).

Inspired by ...

- Our work has been **inspired** by: i) Esteban et al. (2017), a paper on generation of synthetic ICU data; ii) Zhu et al. (2019) on generation of synthetic ECG data of persons with heart diseases; iii) Laptev et al. (2017), a study on anomaly detection with neural networks.

Introduction and Motivation

Research agenda / What are we doing and why?

- Research on **problem gambling** in the **time-series domain** is scarce and is mainly focused on time-series clustering (Suzuki et al., 2019, Peres et al., 2021).
- Our work is a step towards an **early detection of episodes of problem gambling** with neural networks in time series.
- We pursue the following **goals**:
  1. To generate **synthetic time-series of gambling behavior** which can be further used to **i) increase training/test sets** for time-series models; **ii) derive features** for aggregated cross-sectional data; **iii) perform Monte Carlo** studies; **iv) resolve privacy and data protection issues** for replication of research papers.
  2. To apply neural networks for **detection of early signs of problem gambling**.

More on pros and cons of neural networks

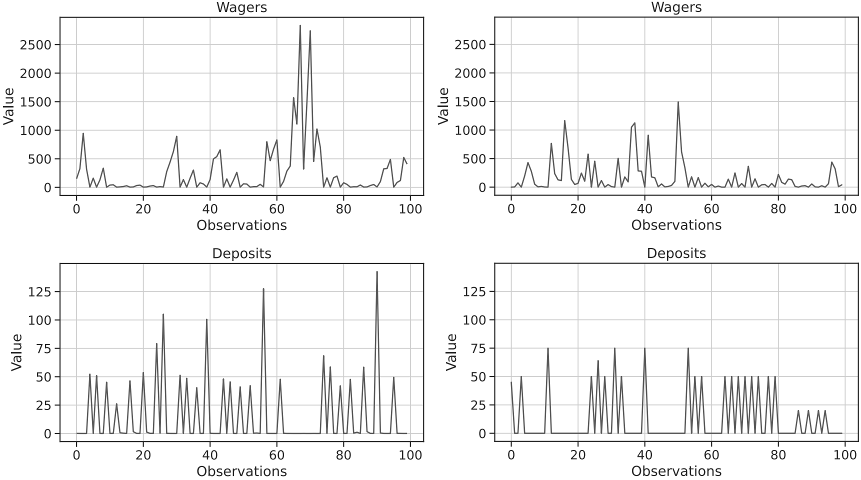

Figure (1) Which one is the GAMbler and which is the GANbler?

Introduction and Motivation

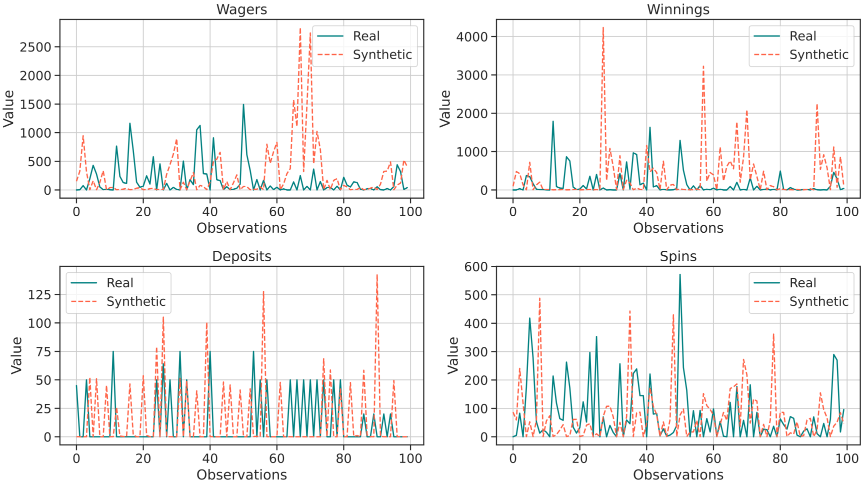

Figure (2) Sample of synthetic and real sequences

Back

Contents

Data

Wagers	Winnings	Deposits	Spins	Array 1	Array 2	Array 3	Array 4	Array 5
.	.	40	.					
4.8	3	.	24					
41.7	53.2	.	139					
.	.	40	.					
1.5	0.3	.	9					
11	2.5	.	43					
37.2	25.41	.	125					
5.4	0.1	.	14					
8	3.6	.	32					
.	.	50	.					
60.1	47.4	.	150					
12.1	16.2	.	54					

Figure (3) Example for sequences with length of ten

- Anonymous transaction data from the **Safe-Server** (for supervision and research purposes) in the German federal state of Schleswig-Holstein between January 2020 and July 2022.
- Merged gambling (wagers, winnings, spins) and non-gambling (deposits) transactions.
- Time-series of **self-excluded** and **provider-excluded** gamblers have been selected maximizing the total number of transactions.
- The data has been **scaled** before being used as inputs for training.

Contents

Generative Adversarial Networks

- The plain vanilla GAN (Goodfellow et al., 2014): two-player minimax game between the **generator** and the **discriminator**.
- The generator is trained using **random noise** to produce realistic **synthetic** samples, which the discriminator would eventually fail to correctly distinguish from the **real** ones.

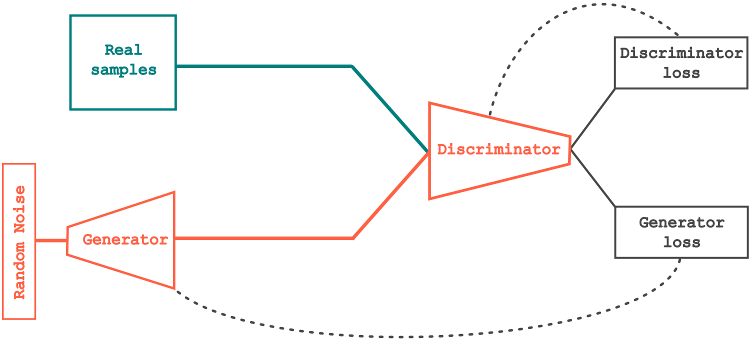

Figure (4) Schematic representation of a vanilla GAN

Generative Adversarial Networks

- Plain vanilla GANs suffer from **mode collapse** and **unstable** training (Brophy et al., 2023).
- The **time GAN** (Yoon et al., 2019), based on Gated Recurrent Units (GRU), is meant to solve these problems by incorporating **embedding** (encoder) and **recovery** (decoder) components to provide mappings between the feature space and the latent code. The latter components form an autoencoder.
- Other notable examples: Esteban et al. (2017), Lin et al. (2020), Ni et al. (2021).

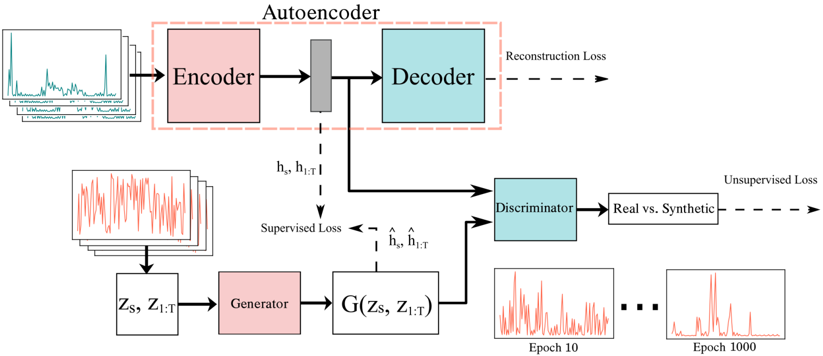

Figure (5) Schematic representation of a time GAN (based on Brophy et al., 2023)

Tuning the time GAN

Tuning time GAN for the gambling data

- Original **time GAN** was tested on autoregressive multivariate gaussian data, cyclical data, stock prices (see also Jansen, 2020), energy consumption and irregular events. Current implementations are known to perform well only on rather short sequences.
- Our changes to the architecture, optimization and training allow us to generate sequences up to 100 of a rather high quality:
  - Training and optimization → **gradient clipping** (Zhang et al., 2019, Esteban et al., 2017) and bias and weight **initialization** to stabilize training (Glorot and Bengio, 2010).
  - Architecture → a **pyramidal architecture** for the generator and discriminator.
  - Generator → an additional **noise layer** added to the generator to improve diversity (Noh et al., 2017).
  - Discriminator → an additional **dropout layer** to improve classification performance (Srivastava et al., 2014).
- Developing environment: Python (TensorFlow, Keras)

More on the architecture

Examples and evaluation

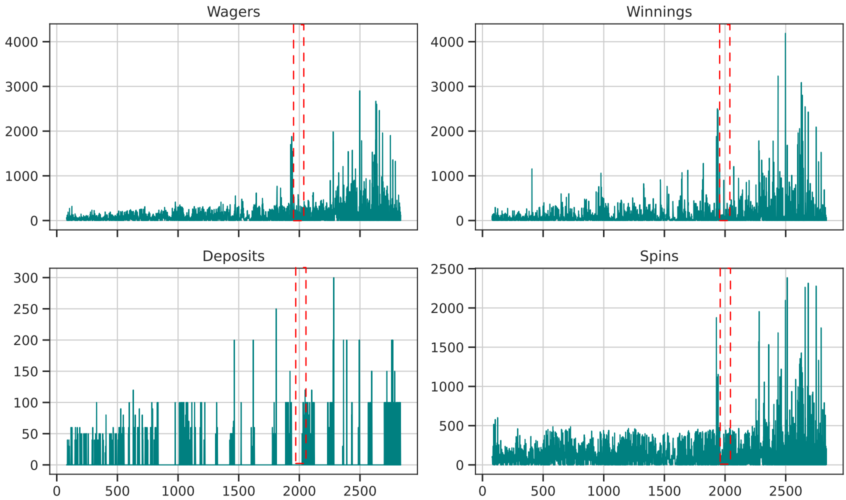

Figure (6) Real time series; sampling window

Examples and evaluation

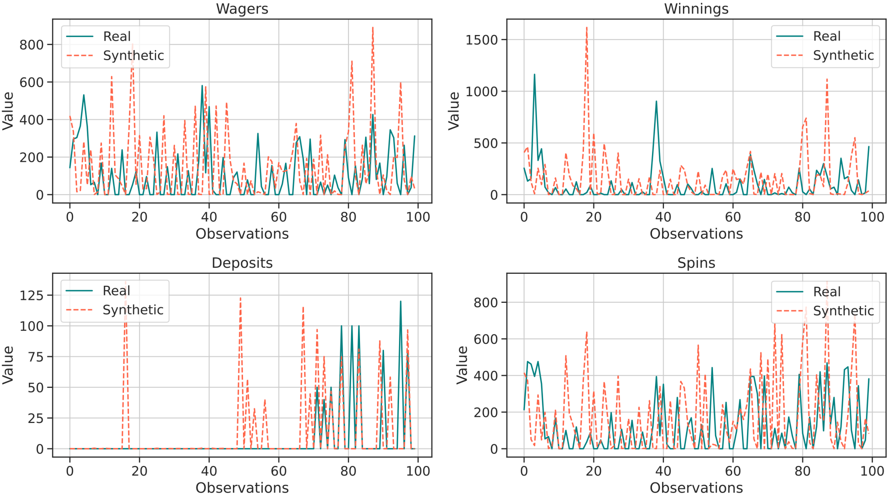

Figure (7) Sample of synthetic and real sequences

Examples and evaluation

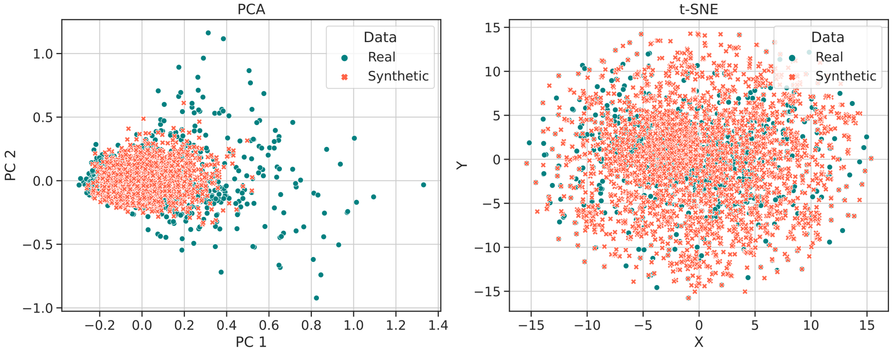

Figure (8) Evaluation of **diversity** (Principal Component Analysis, left panel and t-distributed Stochastic Neighbor Embedding, right panel)

Examples and evaluation

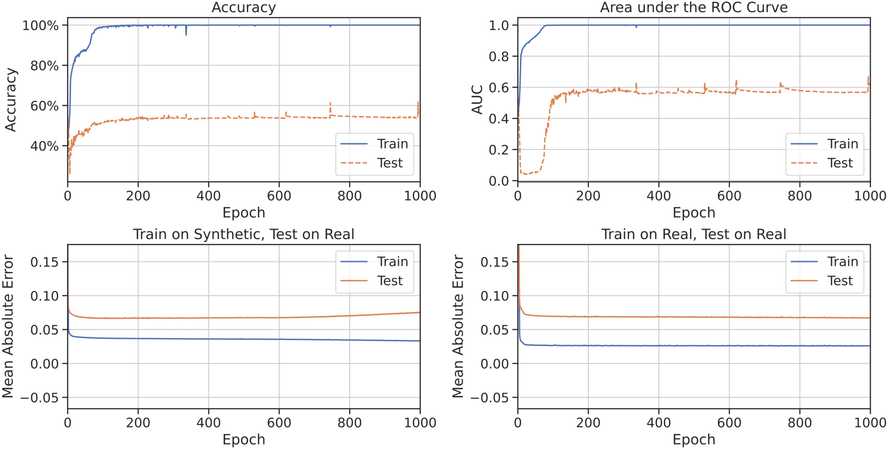

Figure (9) Evaluation of **fidelity** (classification with a GRU classifier; upper panel) and **usefulness** (10 step forecast with a GRU model; lower panel)

Summary

Table (1) Comparison of descriptive statistics

Statistics	Real data	Synthetic data
Mean Wagers	126.25	140.67
Mean Winnings	123.70	139.24
Mean Deposits	75.35	61.45
Mean Spins	153.86	165.25
SD Wagers	159.24	162.74
SD Winnings	209.33	212.43
SD Deposits	32.47	37.98
SD Spins	172.03	183.81

- After aggregation the synthetic time series preserve original properties.
- One of the key challenges is related to reproduction of extreme outliers.

Contents

Multivariate anomaly detection with an autoencoder

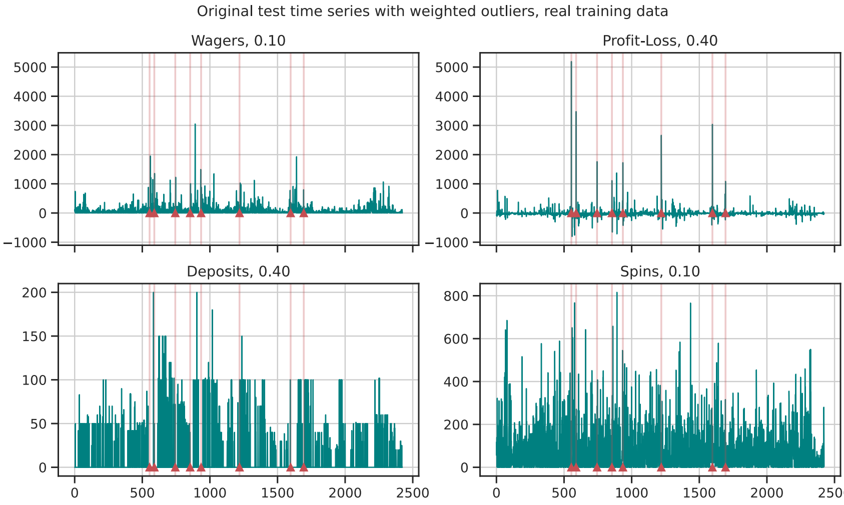

Figure (10) Provider-excluded test sample; weighted outliers; trained on **real** data

Multivariate anomaly detection with an autoencoder

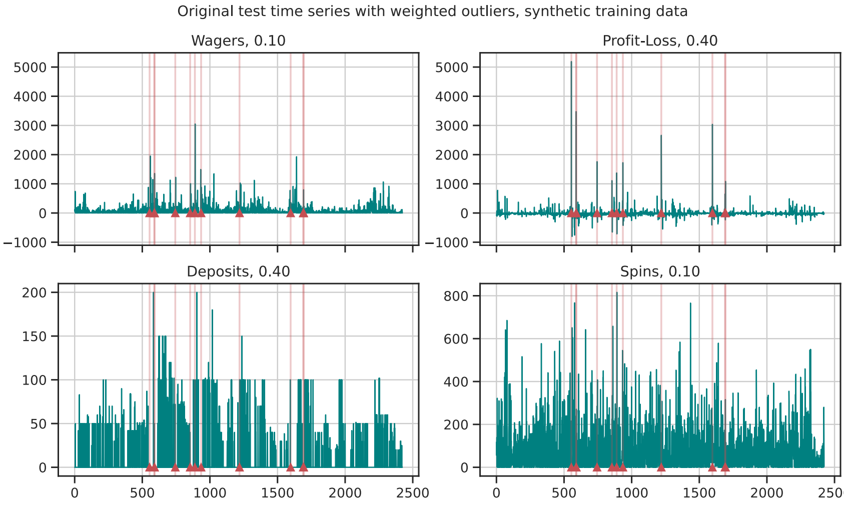

Figure (11) Provider-excluded test sample; weighted outliers; trained on **synthetic** data

Summary

Table (2) Missed opportunities for provider-exclusion measured in days and profit-loss (PL)

Gambler	Days history	To exclusion	To exclusion (%)	PL (EUR)	PL (%)
1	413	286	69.25%	-5157	79.89%
2	394	338	85.79%	-13933	71.67%
3	387	99	25.58%	-2305	68.17%
4	677	91	13.44%	-3577	39.42%
5	814	572	70.27%	-12578	57.61%

- A tuned autoencoder can be used to detect anomalies in gambling behavior as deviations from a forecast.
  More on the architecture
- Training on the real data yields similar list of anomalies as for training on the synthetic data.
- In most cases provider exclusion is initiated too late, after numerous episodes of problem gambling.

Contents

Discussion and conclusion

Discussion

- Potential for bi-directional (recurrent) and multi-directional (convolutional) designs and application of variational autoencoders.
- For time series GANs, higher complexity does not necessarily lead to improvements in fidelity or diversity.
- Desired sequence length is related to data availability.
- Outliers, monotonous and cool down phases may be challenging to replicate.

Conclusion

- A **fine-tuned time GAN** can generate relatively long synthetic sequences representing gambling behavior which partly satisfies diversity, fidelity and usefulness criteria.
- The generated **synthetic data** can potentially be used to **enrich training and test sets**, **derive features**, perform **simulation studies** and enable **replication** of existing research.
- The **autoencoder (trained on real or synthetic data)** can be further tuned for a multivariate **anomaly detection** to reveal the opportunity costs of non-exclusion, or exclusion delays.

Thank you for your attention!
Please do not hesitate to contact us for questions and suggestions!

vadim.kufenko@uni-hohenheim.de
steffen.otterbach@uni-hohenheim.de

Visit us here:
https://gluecksspiel.uni-hohenheim.de/en

References

Brophy, E., Wang, Z., She, Q., and Ward, T. (2023). Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.*, 55(10).

Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. *ArXiv*, abs/1706.02633.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

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

Jansen, S. (2020). *Machine Learning for Algorithmic Trading: Predictive Models to Extract Signals from Market and Alternative Data for Systematic Trading Strategies with Python*. Expert Insight. Packt Publishing.

Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, 83:105662. (IF-2019=4.873).

Laptev, N. P., Yosinski, J., Li, L. E., and Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber.

Lin, Z., Jain, A., Wang, C., Fanti, G., and Sekar, V. (2020). Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 464–483, New York, NY, USA. Association for Computing Machinery.

Ni, H., Szpruch, L., Vidales, M. S., Xiao, B., Wiese, M., and Liao, S. (2021). Sig-wasserstein gans for time series generation. *CoRR*, abs/2111.01207.

Noh, H., You, T., Mun, J., and Han, B. (2017). Regularizing deep neural networks by noise: Its interpretation and optimization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5115–5124, Red Hook, NY, USA. Curran Associates Inc.

Pathare, A., Mangrulkar, R., Suvarna, K., Parekh, A., Thakur, G., and Gawade, A. (2023). Comparison of tabular synthetic data generation techniques using propensity and cluster log metric. *International Journal of Information Management Data Insights*, 3(2):100177.

Peres, F., Fallacara, E., Manzoni, L., Castelli, M., Popovič, A., Rodrigues, M., and Estevens, P. (2021). Time series clustering of online gambling activities for addicted users' detection. *Applied Sciences*, 11(5).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Suzuki, H., Nakamura, R., Inagaki, A., Watanabe, I., and Takagi, T. (2019). Early detection of problem gambling based on behavioral changes using shapelets. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 367–372.

Contents

Introduction and Motivation

Pros of neural networks

- No underlying model required
- Able to learn the smallest details
- Capture path dependence
- Multivariate framework
- Flexible architecture

Cons and Potential risks

- Complex neural architecture search
- Complex diagnostics/evaluation
- Demanding (time and dimensions)
- Mode collapse
- Training instability

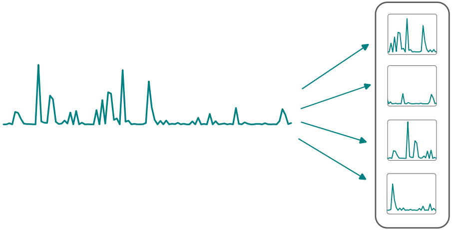

Figure (12) Time series properties as features; univariate example with four neurons

Back

Tuning the time GAN

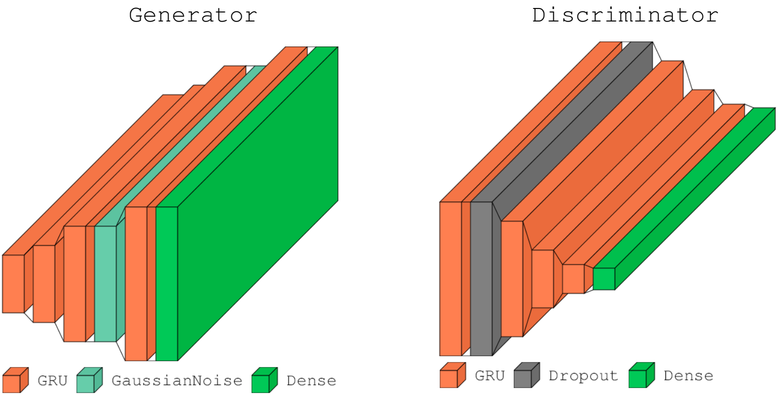

Figure (13) Schematic representation of the tuned time GAN elements

Back

Tuning the autoencoder

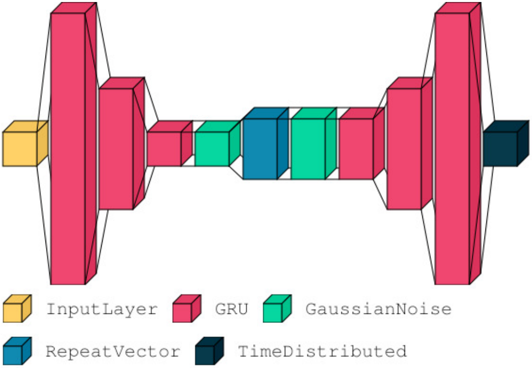

Figure (14) Schematic representation of a tuned autoencoder

Back

Examples and evaluation: white noise test

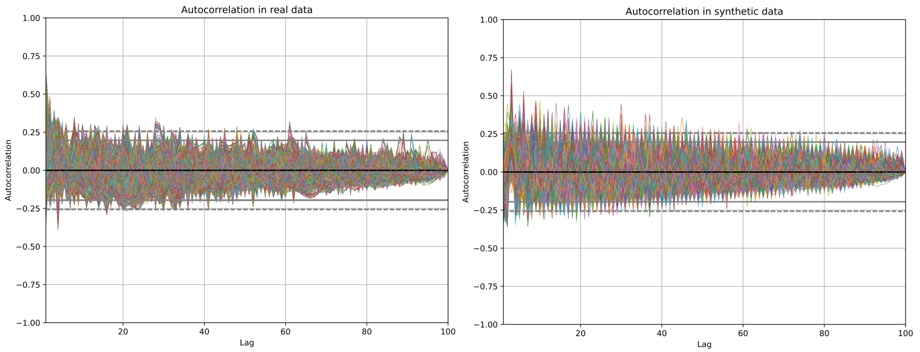

Figure (15) Comparison of autocorrelation functions