Time Series Synthesis using Generative Adversarial Networks

Jan Mark Dannenberg under supervision of Lydia Y. Chen, Aditya Kunar and Zilong Zhao

TU Delft, Computer Science & Engineering

INTRODUCTION

Generative Adversarial Networks [1] for time-series data synthesis.

Research goals:

- Reproducing results for TimeGAN [2].
- Propose improvements for reducing training time and compatibility.

TIMEGAN

Generative Adversarial Networks; **GANs**

- **Generator:** Generates synthetic data
- **Discriminator:** Distinguish between real and fake data from generator.

Generator and discriminator learn by competing against each other; **adversarial loss.**

TimeGAN adds two additional losses:

- **Reconstruction loss:** embedding & recovery function map to and from latent space.
- **Supervised loss:** forces generator to learn temporal dynamics of data.

Adding losses and components adds computational overhead; longer training time. Three training phases. (*Figure 3)*

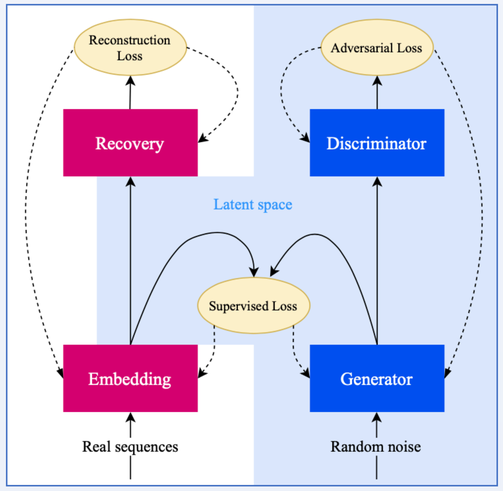

Figure 1: Block diagram of TimeGAN training

REPRODUCING RESULTS

Discriminative score: Measurement of similarity between original and synthetic data.

Predictive score: Measurement to evaluate how well the model learned temporal dynamics of data.

Three data sets: **sine**, **stock** and **energy**.

The original implementation of TimeGAN [4] did not produce good results and no learning curve.

Alternative implementation of YData [3] started **overfitting** after 25000/30000 iterations.

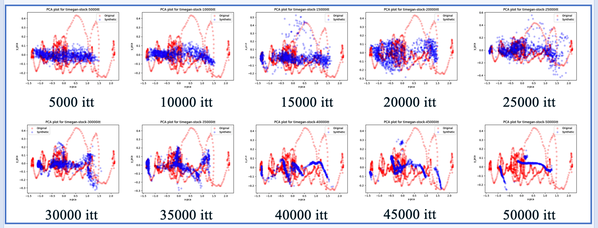

Figure 2: Overfitting in PCA plots every 5000 iterations

Original TimeGAN implementation did not take the **input time information** into account.

After investigation, due to insufficient code review when upgrading to Tensorflow 2.

After fixing the implementation we achieved good results for sine and stock data set.

For the energy we were unable to find correct hyper paramaters.

		Sine	Stock	Energy
Discriminative score (Lower the better)	Our results	0,036	0,141	0,402
	TimeGAN paper	*0,011*	*0,102*	*0,236*
Predictive score (Lower the better)	Our results	0,097	0,039	0,252
	TimeGAN paper	*0,093*	*0,037*	*0,273*

Table 1: Results of reproducing TimeGAN

	Embedding	Supervised	Joint
C1	*72500*	*72500*	*5000*
C2	*70000*	*70000*	*10000*
C3	*67500*	*67500*	*15000*
C4	*65000*	*65000*	*20000*
C5	*62500*	*62500*	*25000*
C6	*60000*	*60000*	*30000*
C7	*57500*	*57500*	*35000*
C8	*55000*	*55000*	*40000*
C9	*52500*	*52500*	*45000*
Benchmark	*50000*	*50000*	*50000*

Table 2: Configurations for distributing iterations

PROPOSED IMPROVEMENTS

Tensorflow 2 compatibility: ensures compatibility across systems and allow for frameworks like Keras.

Re-weighting iterations: improve overall performance and reduce training time.

- ***Observation:*** Joint Phase takes the longest compared to the other phases. (*Figure 5)*
- ***Hypothesis:*** By focussing more on the components that TimeGAN added we can compensate for training the last phase less.
- ***Evaluation:*** Distributing the iterations over the phases of training. (*Table 2)*
- ***Results:*** Configurations C6 to C8 produce constant and better results than the benchmark and reduce overall training time by **9% up to 29%**. (*Figure 4)*
- ***Validation:*** Validated finding on sine data set and found similar results.

Conclusion: Based on computational resources, choose a ratio between 2 : 2 : 1 and 1,4 : 1,4 : 1 to produce equal or better results compared to original 1 : 1 : 1 ratio and reduce training time significantly.

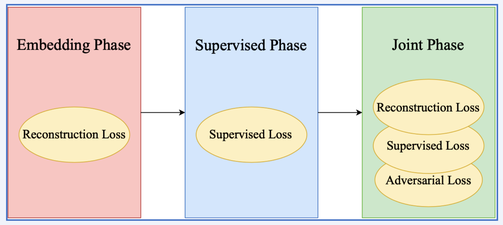

Figure 3: Three phases of TimeGAN training

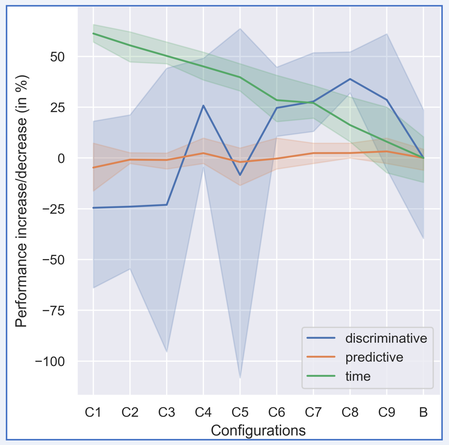

Figure 4: Results for configurations from Table 2

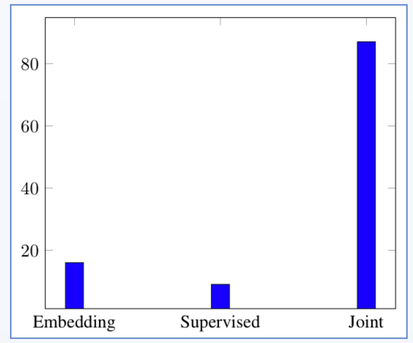

Figure 5: Training times of phases of TimeGAN

CONCLUSION

TimeGAN effectively learns the temporal dynamics of time-series data and is able to generate realistic looking synthetic data.

To ensure reproducibility providing the original implementation and hyper parameters is crucial.

We propose two novel improvements to TimeGAN: compatibility with Tensorflow 2 and distributing the iterations over the training phases.

By re-weighting the iterations we improve the overall performances of TimeGAN and reduce training time.

Further research could be conducted on the impact of the different components of TimeGAN and making TimeGAN differentially private.

REFERENCES

[1] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair, Aaron Courville, Ian J. Goodfellow, Jean Pouget-Abadie and Yoshua Bengio. Generative Adversarial Networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, 2014

[2] Daniel Jarrett Jinsung Yoon and M. V. D. Schaar. Time-series generative adversarial networks. NeurIPS, 2019.

[3] YData AI. YData - YData-synthetic GitHub.

[4] Jinsung Yoon. TimeGAN implementation on GitHub.

CONTACT

{j.m.dannenberg, a.kunar}@student.tudelft.nl,
{y.chen-10, z.zhao-8}@tudelft.nl