Towards Realistic Financial Time Series Generation via Generative Adversarial Learning

1st Mihai Dogariu
University "Politehnica" of Bucharest
Bucharest, Romania
mihai.dogariu@upb.ro

2nd Liviu-Daniel Ştefan
University "Politehnica" of Bucharest
Bucharest, Romania

3rd Bogdan Andrei Boteanu
University "Politehnica" of Bucharest
Bucharest, Romania

4th Claudiu Lamba
Big Data & AI Lab
Hana Institute of Technology, Hana TI
Seoul, South Korea
lambac@hanafn.com

5th Bogdan Ionescu
University "Politehnica" of Bucharest
Bucharest, Romania
bogdan.ionescu@upb.ro

***Abstract*—Training network models to accurately respond to market fluctuations requires access to vast amounts of data. Data availability is strictly bound to the market's evolution, which updates only on a daily basis. In this paper, we propose several solutions based on Generative Adversarial Networks for providing artificially generated time series data with realistic properties. The main challenge here is the specificity of the target data, which has properties that are difficult to control and have wide variations in time, e.g., central moment statistics, autocorrelation or cluster volatility. Another contribution is in assessing the quality of synthetic data, in general, as there is no standard metric for this. Experimental validation is carried out on real-world financial data retrieved from the US stock market.**

***Index Terms*—financial, time series, fintech, adversarial, GAN, generation**

I. Introduction

Financial markets have always been a focus point for researchers around the world because of the major financial gain that accurate insights would provide. Scientists have tried to design models that fit the markets in an attempt to make reasonable analysis and automatic prediction of various phenomena. In practice, two major directions have been explored: (i) use of stochastic processes [1], [2], and (ii) employment of agent-based models [3], [4]. Nevertheless, financial time series proved to be too complex to be modeled perfectly with these mathematical tools [5]. A solution are the current state-of-the-art deep neural networks. To be able to exploit their full potential in providing accurate predictions, they require even more data than the traditional handcrafted feature-based learning models. Thus, it is of great interest to be able to synthesize new financial data which resembles real stock markets, on the spot, to train complex models.

One approach is to focus on predicting the next sample(s) in a sequence based on the available recent history. This can be modeled as trying to perform the prediction $p(x_{t+1}, ..., x_{t+n}|x_1, ..., x_t)$, where $x_1, ..., x_t$ are the historical values from moment 1 up to moment t and $x_{t+1}, ..., x_{t+n}$ are the future n samples that are being predicted. Most of the time, $n = 1$ is considered to be enough. For instance, Zhou et al. [6] propose a generic framework employing Long Short-Term Memory (LSTM) and convolutional neural networks for adversarial training to forecast high-frequency stock market. Wiese et al. [7] use temporal convolutional networks and manage to capture longer-ranging dependencies such as the presence of volatility clusters. A more general approach is TimeGAN [8], a framework that learns an embedding space jointly optimized with both supervised and adversarial objectives. Kim et al. [9] tackle financial time series prediction with stacked LSTM, attention networks, and weighted attention networks.

Another approach is to generate a fixed number of consecutive samples at a time, in a data set augmentation manner. This procedure requires to extract windows of a given length L_w when creating the training data set for the GAN. An example is the framework in [5], where the authors propose several mixtures of multilayer perceptrons and convolutional neural networks to generate fixed-length time series.

In this paper, we push forward the advances in accurate window-based time series data generation and bring the following contributions: (i) we adapted various network architectures commonly used for images to 1D time series generation, (ii) we are the first to propose a multi-channel approach for stock prices generation, (iii) we approach the evaluation of the quality of the synthetic generated data and propose qualitative, and quantitative measurements. Although our work follows a similar principle to Takahashi et al. [5], we explore more advanced network models, introducing a novel multi-channel perspective, as well as an extensive evaluation.

The remainder of the paper is organized as follows. Section II provides a description of the financial time series. The proposed solutions are presented in Section III. The experimental setup and results are presented in Section IV. Finally, Section V provides the conclusions and discusses future work.

This paper is the result of research funded by Hana TI.

II. Financial time series

A company's financial time series represents the chronological evolution of several indicators. To create good prediction models it is necessary to train on sufficiently large data sets. However, this resource is not always freely available and, when it is, it does not provide enough data for more complex models. Generating realistic synthetic data is therefore a valuable lead.

In our work, we address the Open - High - Low - Close (OHLC) indicators which are critical for market decisions, e.g., the closing price captures information about how well or poorly a stock performed during the day. In particular, we investigate log returns, i.e., the logarithm of ratios of closing prices from consecutive days, given by $r_i = \ln \frac{C_i}{C_{i-1}}$, where C_i represents the closing price of day i and r_i the log return closing price of day i. This ratio is useful because it reduces not only the intra-variation of the time series, but also acts as a normalization between different companies' stocks, as displayed in Figure 1. We can see there that two stocks that do not have the same behavior and whose magnitudes of the closing prices belong to different ranges can be successfully encoded under the same range by applying the log return transformation.

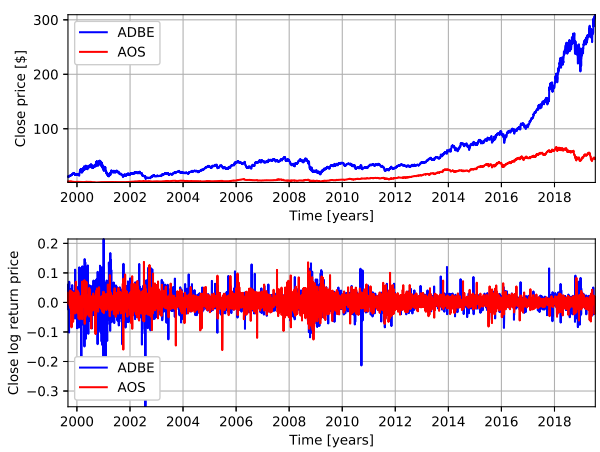

Fig. 1. Log return influence on the intra and inter variations of the closing prices for two companies (A. O. Smith Corp — AOS and Adobe Inc. — ADBE); top — closing prices evolution over 20 years, bottom — log return of the same prices.

Challenge. Unlike other data, financial time series acquisition cannot be hasted since it is necessary to wait for an entire day to extract one new sample, given that closing prices are updated at the end of the trading day. Furthermore, the microstructure of the financial market gives the financial time series several properties and shapes [7], [10], [11]. It is known that these data are more peaked than normal distributions and exhibit a fat-tailed behavior (extreme values are more likely to occur). Also, large changes of prices tend to cluster together, an effect called volatility clustering. Lastly, empirical asset returns are uncorrelated for any value of the lag larger than one, but not independent. Therefore, generative models should deal in particular with these aspects.

III. Proposed approaches

Our solution is to synthesize large windows of data via GANs. Unlike [6]–[9], we do not aim to generate the $(n+1)^{th}$ sample, given the previous n samples, but we want to generate entire new 1-dimensional arrays of length n. The target is to generate a fixed-length vector of log return Close prices. The only work similar to ours that we could find in the literature is of Takahashi et al. [5]. Therefore, we recreated and adapted the architectures proposed by them as a baseline. Since an exact description of the layers that they used is missing, we experimented with different setups and reported only the best performers.

In the following, we describe each of the architectures that were implemented. The length of the synthesized 1D array, L_w, has been set to 250 for all models, the equivalent of an entire working year in finance. Please note that all architectures are presented in their optimized versions, achieved after in-depth ablation studies.

Fully connected GANs - MLP. We implemented four MLP architectures with different number of layers and neurons per layer, denoted MLP_1 to MLP_4.

MLP_1's generator is a $20 - 40 - 80 - 160 - L_w$ network and the discriminator has a $L_w - 80 - 40 - 20 - 10 - 1$ structure, where each value represents the number of neurons in the respective layer. We used Leaky ReLU activations [12] and dropout layers in the Generator with batch normalization layers [13] in the discriminator. MLP_2 has the same layer dimensions as MLP_1, but without the dropout and batch normalization layers. MLP_1 and MLP_2 are adapted from Takahashi et al. [5]. Additionally, we propose MLP_3 which has a $20 - L_w - L_w - L_w - L_w$ generator, and a $L_2 - 80 - 80 - 20 - 20 - 1$ discriminator, without dropout and batch normalization. We also implemented MLP_4, a shallow version of MLP_3, with only the first 3 layers in the generator and discriminator in order to assess the impact that the number of layers has.

Fully convolutional GANs - FCGAN. We implemented three architectures for fully convolutional GANs: $FCGAN_1$ to $FCGAN_3$, which were derived from DCGAN [14].

All models follow a $100 - 96 - 48 - 24 - 12 - 1 - fc(L_w)$ structure in the generator and a $L_w - 1 - 12 - 24 - 48 - 96 - 1 - fc(1)$ structure in the discriminator, where each value represents the number of feature maps in the respective layer and $fc(x)$ denotes a fully connected layer, where x is the size of the output array for each channel. $FCGAN_1$ uses transpose convolutions followed by batch normalization layers and ReLU activations in the generator. The discriminator has a mirrored structure, with convolutional layers replacing the transpose convolutions and Leaky ReLU replacing ReLU activations; $FCGAN_2$ is same as $FCGAN_1$, but without any batch normalization; $FCGAN_3$ is a shallow version of $FCGAN_1$, replacing the last 2 layers in both generator and discriminator with a flattening layer to adapt to the output dimension. Both $FCGAN_1$ and $FCGAN_2$ are adaptations of the models proposed by Takahashi et al. [5]. Again, since an

accurate description of the layers' setup is missing we tried several models and reported the best ones.

Multichannel GANs - FCGANmc. We propose a new multichannel approach, denoted $FCGANmc$. To the best of our knowledge, this is the first time it is used in the literature. Different from previous approaches, we intend to generate entire OHLC financial time series (see Section II). The generator has a $100-96-48-24-12-4-fc(L_w)$ structure, and we used a $L_w-1-12-24-48-96-4-fc(1)$ setup for the discriminator. The fully connected layer in the end is still necessary to switch from the dimension of each feature map to the dimension of the generated sample. More information on this model is given in subsection IV-A.

Spectral normalization GAN - snFCGAN. We use a $100-96-48-24-12-1-fc(L_w)$ structure in the generator and a $L_w-1-12-24-48-96-1-fc(1)$ structure in the discriminator, but with spectral normalization layers replacing batch normalization layers. We are interested to see spectral normalization's impact since this technique has been proven in the literature [15] to offer better results for image generation by avoiding the notorious mode collapse.

Wasserstein GANs - WFCGAN and WMLP. We propose a Wasserstein GAN [16] implementation, with weight clipping, for the MLP_3, $FCGAN_1$ and $FCGAN_2$ architectures, denoted $WMLP_3$, $WFCGAN_1$ and $WFCGAN_2$, respectively. The general architectures remained unchanged, but the optimization was changed to the Wasserstein framework. During each epoch we trained the Discriminator for 5 iterations, then the Generator for one iteration.

IV. Experimental setup

We trained the generator to output 1-dimensional arrays of fixed length or, in the case of the multichannel architectures, 4-dimensional arrays, of length L_w. The generators start from randomly generated noise vectors of size 20 and 100 for the MLP and FCGAN architectures, respectively. All discriminators take arrays of length L_w as input and output a single real value, representing the decision, whether the processed sample was real or not, following the GAN framework. We trained our systems for 200 epochs with fixed learning rates of values from the $\{1e-05, 5e-04, 1e-04\}$ set. We ran all our experiments using PyTorch [17] on a system with 2 NVIDIA QUADRO M4000 GPUs.

A. Data

For training the models, we used the S&P index[1], provided by Hana Institute of Technology. This extended data set consists of 1,506 companies with daily OHLC records from January 1st 1999 to June 17th 2019.

The data set consists of time series with different starting dates (the date when the respective company was first listed on the market), but the same end-date, namely the last day of the data set. The nature of the financial data gives rise to the problem of training a system to generate fixed length samples on a data set with varying-length instances. The solution that we found for this problem was to split each available time series into chunks of a fixed number of samples using a sliding window. For each company we slid a window of length $L_w = 250$ days across its closing log returns vector, with a stride of 30 days. This means that we extract one year's worth of data for each window, with a stride of 6 weeks, expressed in trading days, and add each such window to our training set. The discriminator extracts random batches from this set to compare with the samples synthesized by the generator.

For the multichannel architectures, we generated OHLC using four channels instead of one in the fully convolutional setup. OHLC values may be several orders of magnitude different between companies, but they are close to each other for the same company. Also, daily updates bring only small absolute modifications for any of these 4 prices. Therefore, it is easier to encode ratios of OHLC values, rather than their absolute values. Thus, instead of Open prices we used $\frac{Close_n - Open_n}{Close_{n-1}}$. For the next 3 channels we imposed a restriction on the model to generate only positive values. This makes it easier for the model to infer more relevant samples. Given that High represents the maximum value that the stock took that day, we created a vector of $\frac{High_n - Close_n}{Close_{n-1}}$, which are sure to be positive values. Similarly, we generated a vector of $\frac{Close_n - Low_n}{Close_{n-1}}$, which results also in positive values. The last channel is represented by the Close return prices, i.e. $\frac{Close_n}{Close_{n-1}}$. For Open prices, however, this logic couldn't be applied, since Open prices might be greater or lower than Close prices, without any particular rule.

B. Evaluation

While the evaluation of classification and retrieval systems is a well know problem and many metrics were validated during time [18], the evaluation of synthetic generated data is still an open issue, especially in the financial field. Several attempts have been made to assess the goodness of synthesized images, e.g., Inception Score [19] and Fréchet Inception Distance [20] but transforming financial time series to images behaves poorly when assessed by a neural network trained on natural images. We therefore analyze and propose a series of metrics that were inspired by signal processing problems. Most of these metrics are still experimental regarding how accurate they can describe the performance of a financial data GAN, but they can still help establish an hierarchy between different models. We approach the evaluation at two levels: (i) qualitatively, and (ii) quantitatively, as presented in the following.

Qualitative analysis. Lucic et al. [21] argue that it is necessary to report a summary of distribution of results, rather than a single best result achieved. To capture as much information as possible, we randomly select one batch of real data and one of synthesized data during each epoch and we compare their distributions. In order to check whether the synthesized samples are coherent from a financial point of view, we propose to explore the following properties: *Central moments* — two distributions that share the same behavior for mean, standard deviation, skew and kurtosis are likely to be similar;

[1]https://www.spglobal.com/ratings/en

TABLE I
QUALITATIVE AND QUANTITATIVE RESULTS (✓MEANS THAT THE PROPERTY IS MET, WHEREAS ✗THE CONTRARY).

Arch. name	Mean	STD	Skew	Kurtosis	Autocorr	Heavy-tail	Cluster Volatility	t-SNE	D_{KL}	JSD	K-S statistics	EM distance
MLP_1 [5]	✗	✗	✓	✗	✗	✗	✗	✗	721.5	0.5757	0.4021	0.1140
MLP_2 [5]	✓	✓	✓	✗	✓	✓	✗	✗	29.81	0.0897	0.0301	0.0028
MLP_3	✓	✓	✓	✗	✗	✓	✗	✗	128	0.1235	0.0465	0.0088
MLP_4	✓	✓	✓	✗	✗	✗	✗	✓	99.89	0.2095	0.131	0.0106
$WMLP_3$	✓	✓	✓	✓	✓	✓	✗	✗	39.08	0.1031	0.0323	0.0030
$FCGAN_1$ [5]	✗	✗	✗	✗	✗	✗	✗	✗	197.5	0.2315	0.1709	0.0115
$FCGAN_2$ [5]	✓	✓	✓	✓	✗	✓	✗	✓	13.26	0.0454	0.0178	0.0011
$FCGAN_3$	✗	✗	✗	✗	✗	✗	✗	✗	673.8	0.5341	0.4007	0.1026
$WFCGAN_1$	✓	✓	✓	✗	✓	✓	✗	✓	18.23	0.0570	0.0359	0.0018
$WFCGAN_2$	✓	✓	✗	✗	✗	✓	✓	✓	26.06	0.0825	0.0387	0.0034
sn_FCGAN	✗	✗	✗	✗	✗	✓	✗	✗	23.53	0.0953	0.0408	0.0031
$FCGANmc$	✓	✓	✓	✓	✗	✓	✗	✗	78.16	0.1101	0.0797	0.0032

Autocorrelation — a well-known property of financial time series is that they do not posses a linear predictability, meaning that the autocorrelation of the returns is a diminishing function. *Heavy-tailed distribution* — financial time series are known to exhibit a heavy-tailed behavior, i.e., their distribution presents a higher probability of sampling very high and very low values than a normal distribution would. *Volatility clustering* — another well-known property is the volatility clustering, meaning that changes (either high or low) come grouped in clusters. *t-SNE and PCA for 2D visualization* — these are two common methods to assess the similarity between several data distributions. As t-SNE [22] is computationally heavy to perform on large amounts of data, we follow the authors' advice and first perform a PCA dimensionality reduction for our data.

Quantitative analysis. Quantitative analysis is generally quite difficult to perform on generative models, especially in the context of this relatively new issue of data generation. To provide a solution, we adapt metrics that are generally used in the information theory field as well as metrics designed for generative models but in other fields, such as image and speech processing, namely: Kullback-Leibler divergence, Jensen-Shanon divergence, Kolmogorov-Smirnov test statistics, and Earth Mover's distance [23]. We are interested in the lowest values for each of these metrics. All of them are in accordance with the optimization criterion of GANs and we found them to be most adequate for our setup from an information theoretic point of view.

Results and discussion. As there are very few other approaches tailored for this type of time series data generation in the literature, we compare our proposed architectures to the ones in [5]. Extensive experimenting was conducted varying different parameters. We made snapshots of each network setting whenever it would encounter a new best value for any of the proposed quantitative metrics. Afterwards, we went through all of these snapshots and manually inspected all the qualitative metrics. Empiric results show that among the proposed metrics, the Jensen-Shanon divergence proves to be the best suited for our financial time series GANs, so we present the results for the snapshots that achieved the best JSD for each network model. Table I synthesizes the achieved results.

The first observation is that the ranking based on one metric does not necessarily generalize to other metrics. For the time being, it is safe to assume that if we fix an architecture, the snapshot with the lowest JSD values has overall better performance.

Secondly, by examining the difference between $MLPs$ and $FCGANs$ we observed that it is easier for $MLPs$ to fit the first three central moments than it is for $FCGANs$. However, none of the vanilla $MLPs$ managed to also imitate the kurtosis behavior, in contrast to $FCGAN_2$. Several works [24], [25] insisted on replicating higher order statistics, but their research was carried on the encoded features, rather than on generated samples, which is more suited to autoencoder frameworks (stacked or variational). In [26], the authors show that it is difficult to achieve the third and fourth moment statistics with regular GANs.

Another important aspect refers to the depth of the network. Shallow versions such as MLP_4 and $FCGAN_3$ underperform when compared to their deeper versions, i.e. MLP_3 and $FCGAN_1$, respectively. Increasing the depth of the network further also increases the risk of overfitting due to the large number of trainable parameters that it adds.

We also noticed that batch normalization is drastically decreasing the performance and leads to mode collapse. This was also confirmed by the authors in [5], who also discovered that the addition of batch normalization layers hinders the realistic generation of financial time series. This is counter-intuitive since batch normalization has been successfully used to address this issue, in particular, in the literature.

A surprising result is in the case of the $snFCGAN$, which failed in all qualitative assessments, but obtained very good results for the quantitative metrics. This strengthens more the assumption that there is no certainty that one metric is enough to characterize these type of models.

Our proposed approaches performed generally better than the ones existing in the literature. Concretely, the Wasserstein GANs topped their vanilla counterparts, as we observed an improvement in the case of $WMLP_3$ and $WFCGAN_1$. A

careful tuning of the number of iterations for the critic and the discriminator has to be performed in order to avoid overfitting. Our proposed multichannel architecture managed to attain similar skewness and kurtosis values to the real data, unlike most of the other architectures that we tried. This proves that using more data at each point helps stabilize the training process and attain the desired central moments.

We found that autocorrelation and cluster volatility were difficult to capture in the synthetic data generation. One reason is that the training data contains a great deal of variety. Lastly, the t-SNE visualization was not very informative. We gave a positive vote only in those cases where the data distributions were clearly grouped together and had a similar pattern. Cases which presented outliers from the cluster were, thus, excluded.

V. Conclusions and future work

In this paper, we proposed various GAN architectures for the generation of realistic financial time series. We analyzed the influence of the network parameters, such as the type and number of layers, the underlying framework, and the number of channels. We investigated several quantitative and qualitative metrics that can be useful when using such systems. Experimenting on real-world data, results show that the best candidate architecture is Wasserstein MLP for replicating the statistics of interest whereas the Jensen-Shanon divergence is best suited for determining the model that best replicates the real data probability distribution. We found also that the large variety in the training data set raises numerous problems during the training procedure. One way of overcoming this setback would be to cluster the input data in groups that exhibit similar properties. Overall, the results are very promising and support the possibility of achieving high quality artificial data.

Future work will mainly focus on the investigation of several other more complex generative architectures that involve variational autoencoders (VAEs), VAE-GANs and conditional GANs. Also, other metrics should be explored and a lead is to assess the quality of the prediction for models trained on real and artificial data.

Acknowledgment

Mihai Dogariu, Liviu-Daniel Ştefan and Bogdan Ionescu work has been partly funded by the Ministry of Innovation and Research, UEFISCDI, project SMARTRetail, agreement PN-III-P2-2.1-PTE-2019-0055, 2020-2022.

References

[1] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation," Econometrica, vol. 50, no. 4, pp. 987–1007, 1982.

[2] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," Journal of Econometrics, vol. 31, no. 3, pp. 307 – 327, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304407686900631

[3] D. Challet and Y.-C. Zhang, "Emergence of cooperation and organization in an evolutionary game," Physica A: Statistical Mechanics and its Applications, vol. 246, no. 3, pp. 407 – 418, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437197004196

[4] T. Lux and M. Marchesi, "Scaling and criticality in a stochastic multi-agent model of a financial market," Nature, vol. 397, no. 6719, pp. 498–500, 1999.

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

[6] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, "Stock market prediction on high-frequency data using generative adversarial nets," Mathematical Problems in Engineering, vol. 2018, 2018.

[7] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer, "Quant gans: deep generation of financial time series," 2019.

[8] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in Advances in Neural Information Processing Systems, 2019, pp. 5509–5519.

[9] S. Kim and M. Kang, "Financial series prediction using attention lstm," 2019.

[10] A. Chakraborti, I. M. Toke, M. Patriarca, and F. Abergel, "Econophysics review: I. empirical facts," Quantitative Finance, vol. 11, no. 7, pp. 991–1012, 2011.

[11] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," Quantitative Finance, vol. 1, no. 2, pp. 223–236, 2001. [Online]. Available: https://doi.org/10.1080/713665670

[12] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in Proc. icml, vol. 30, no. 1, 2013, p. 3.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

[14] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015.

[15] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018.

[16] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in Proceedings of the 34th International Conference on Machine Learning - Volume 70, ser. ICML'17. JMLR.org, 2017, p. 214–223.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[18] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to information retrieval. Cambridge university press, 2008.

[19] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2234–2242. [Online]. Available: http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf

[20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in Advances in neural information processing systems, 2017, pp. 6626–6637.

[21] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," in Advances in neural information processing systems, 2018, pp. 700–709.

[22] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," Journal of machine learning research, vol. 9, no. Nov, pp. 2579–2605, 2008.

[23] M. M. Deza and E. Deza, "Encyclopedia of distances," in Encyclopedia of distances. Springer, 2009, pp. 1–583.

[24] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in International Conference on Machine Learning, 2015, pp. 1718–1727.

[25] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," in Advances in Neural Information Processing Systems, 2017, pp. 2203–2213.

[26] J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, H. Xiao et al., "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems," Journal of Computational Physics, vol. 406, p. 109209, 2020.