Hindawi
Scientific Programming
Volume 2021, Article ID 7877590, 13 pages
https://doi.org/10.1155/2021/7877590

Research Article

DAuGAN: An Approach for Augmenting Time Series Imbalanced Datasets via Latent Space Sampling Using Adversarial Techniques

Andrei Bratu and Gabriela Czibula

Faculty of Mathematics and Computer Science Babeș-Bolyai University, Cluj-Napoca, Romania

Correspondence should be addressed to Andrei Bratu; bratuandrei0@gmail.com

Received 7 July 2021; Accepted 14 September 2021; Published 12 October 2021

Academic Editor: Sze-Teng Liong

Data augmentation is a commonly used technique in data science for improving the robustness and performance of machine learning models. The purpose of the paper is to study the feasibility of generating synthetic data points of temporal nature towards this end. A general approach named DAuGAN (Data Augmentation using Generative Adversarial Networks) is presented for identifying poorly represented sections of a time series, studying the synthesis and integration of new data points, and performance improvement on a benchmark machine learning model. The problem is studied and applied in the domain of algorithmic trading, whose constraints are presented and taken into consideration. The experimental results highlight an improvement in performance on a benchmark reinforcement learning agent trained on a dataset enhanced with DAuGAN to trade a financial instrument.

1. Introduction

Data augmentation is a vast and often used method for enhancing the amount of data available for training a *machine learning* (ML) model. It is well known that the amount and quality of data available are closely bounded to the success of any ML project, independent of the application domain. There are multiple data augmentation procedures, which are often specific to the application domain and the specific dataset that is used.

For example, image-based machine learning tasks often employ operations of contrast adjustment, flipping, translation, cropping, rotation, color space manipulation, etc. These present new contexts to the model, helping it to better generalize and to avoid overfitting [1]. Another example of data augmentation is the SMOTE, or the Synthetic Minority Oversampling Technique, whose purpose is to alter the dataset presented to the algorithm by presenting minority-class data points to the classifier more often than they naturally occur (oversampling), while minimising the rate at which the majority class appears (undersampling). This increases the sensibility of the model for the subrepresented data class and has applications such as identifying fraud credit card transactions [2].

A more recent augmentation method involves using a *generative adversarial neural network* (GAN) architecture, whose ability to reproduce a statistical distribution is repurposed for creating new, convincing examples of a poorly represented class, or generally any point of the dataset [3]. GANs are an important machine learning paradigm. Two neural networks engage in a zero-sum game where the generator network attempts to generate new samples, while the discriminator discerns between real samples and generated samples. The end goal is to train the generator into reproducing the initial train distribution as close as possible. GANs have been used to great effect, with examples such as reproducing the effects of dark matter on astronomical observations [4], generating photorealistic human faces [5], or applying style transfer operations in the audiodomain [6].

Identifying imbalanced classes and enhancing their presence in the time series would not be devoid of practical applications. One such example is *securities trading*. Securities are defined as any financial instrument that can be bought or sold via an accredited intermediary, creating a

supply and demand market. An example is the stock market, which allows buying and selling "shares": discreet units of ownership in a company. While the price of any share has a correlation with the business performance, there is research that indicates the market's sentiment and domain-specific factors such as national interest rate create a cyclical effect on the price evolution [7, 8].

The current paper is based on work originating from two research questions:

(1) RQ1. Is it possible to improve the performance of *reinforcement learning-* (RL-) based trading algorithms through augmenting training data using adversarial techniques?

(2) RQ2. What is the performance gain of the RL agent trained on the enhanced data over a baseline RL agent trained on the initial data, without augmentation?

To this end, a general approach named *Data Augmentation using GANs* is proposed for identifying poorly represented sections of security-related time series. Leveraging that autoencoder neural network architectures can encode and decode complex temporal dependencies to and from a latent space, the proposal reduces the problem of identifying poorly represented time series periods into a clustering problem [9] and synthesises new examples of the minority class using a GAN architecture. A method of integrating synthesised data points into the original time series, respecting the original constraints of the data, is presented, and experiments are carried out for measuring the performance improvement on benchmark reinforcement learning algorithms.

Parallels between the proposed method and the TimeGAN method proposed by Yoon et al. for generating arbitrary time-series [10] are acknowledged. The proposed method improves by particularising the problem to the constraints of security-related time series on one hand and the problem of data series minority data augmentation on the other one.

Section 2 introduces fundamental concepts used by the approach, along with a literature review on time series generation. DAuGAN is introduced in Section 3 along with proposed methodology, whilst experimental results and their analysis are presented in Section 4. The conclusions of the paper and directions to further improve and extend DAuGAN are outlined in Section 5.

2. Background

This section presents a literature overview of the technical notions used in this paper. The importance and evolution of generative adversarial and autoencoder networks are presented, together with a brief review on reinforcement learning. Historical approaches related to data augmentation and data synthesis on time series are also presented.

2.1. Generative Adversarial Networks. GANs is a deep learning architecture that have been first introduced by Ian Goodfellow et al. [11], which has been heavily used in image-based tasks, from synthesising new images [12] to reproducing the content of one image in the style of another.

At a very high level, the generative adversarial network technique pits two deep neural networks against each other in a zero-sum game. One of the networks, the *generator*, acts as a map from a latent space towards a desired distribution, sampling noise from the latent space that is synthesised as closely as possible to a point in the distribution. Its counterpart, the *critic*, is fed samples from both the real distribution and from the generator, with the goal of deciding whether the sample is "real" or "fake." Over time, the two networks improve at their goal, resulting in not only better fakes from the generator but also a better ability to discern the fakes from the critic. Ideally, the system converges towards equilibrium where the critic can no longer separate between the two classes; i.e., it assign equal probability for any sample to be either one of the classes.

Formally, the generator attempts to minimise the value of the following loss function, while the discriminator attempts to maximise it: $E_x[\log(\mathrm{D}(x))] + E_z[\log(1 - \mathrm{D}(G(z)))]]$, where x is a random variable corresponding to the real distribution, z is a random variable assigned to the generated distribution, $G(x)$ is the generator's output, $D(x)$ is the discriminator's output, and E_v denotes the expected value over all instances v.

The expected value is used to indicate that the loss is the average over all samples of the batch at a given training step. The critic assigns values from 0 to 1, estimating the probability that a sample is real. Letting x denote the real distribution and z denote the synthesised distribution, the critic attempts to maximise this loss, the upper bound being obtained when all labels are properly assigned, while the generator attempts to minimise it by controlling the second term, i.e., generating more convincing examples, signified by the $G(z)$ term. The log operations are derived from the cross entropy between the real and fake distributions.

Notorious problems affecting GANs are *mode collapse* and *vanishing gradients*. *Vanishing gradients* is a general issue in machine learning, where gradients prove insufficient for the machine learning model to update meaningfully, while this issue has classically occurred in neural models with high depth [13] or recurrent networks such as long-short term memory architecture [14]. However, the issue manifests particularly in the case of generative adversarial networks: the unadjusted loss formulation presented above will result in a critic that converges faster than the generator. Thus, the critic cannot offer constructive feedback for the generator to improve on, since it perfectly discerns between real and fake.

A connected issue with vanishing gradients that is faced by generative adversarial networks is *mode collapse*. The problem manifests on the generator's side by mapping all latent points to the same synthesised sample. From the perspective of game theory, both *mode collapse* and *vanishing gradients* issues are caused by the two players, the critic and the generator, converging to a local, undesired optimum of the game that does not offer enough incentives for any of the networks to update their weights [15].

Several improvements on the domain transfer subproblem have been addressing the mode collapse issue. CycleGAN [16] introduces the following adjustments: instead of sampling from a latent random space, the generator samples from the input space of the input domain, with output in the target domain. The critic is fed both generated images and those belonging to the target domain, thus encouraging the generator to learn a better mapping between input and target. Thus, the improvement resides in translating the task into an unsupervised task, as the generator is ideally able to map any image from the first domain to an image in the target domain. A limitation of the CycleGAN paper is the domains being required to be homogeneous [17]. An improvement over the CycleGAN is represented by TraVeLGAN [18], which adds to the classical two network architectures formed of a generator and critic, a third, siamese network, and eliminates the domain homogeneity constraint [19].

The Wasserstein variation of the generative adversarial network architecture (WGAN), authored by Arjovsky et al. [20] offers a robust method to train GAN architectures. The WGAN improves the stability of learning, eliminates problems such as mode collapse, and provides meaningful learning curves useful for hyperparameter searches.

2.2. Autoencoders. The autoencoder (AE) architecture uses a two-part neural network to transform the input in a compressed and meaningful representation using the encoder part, recreating the input using the decoder part [21]. The technique proves to be immensely flexible and is of interest to the purpose of this thesis since former research proved that autoencoders are able to encode and decode complex temporal features into the latent space [9]. Furthermore, there are no special theoretical aspects to be considered over a general-purpose deep learning architecture. The autoencoder is presented as two symmetrical parts, with a small, latent representation in the middle, usually trained to minimise the mean squared error between the distribution and itself.

AEs can be interpreted as an improvement over the statistical technique of principal component analysis. Principal component analysis with p dimensions identifies an orthonormal base of p vectors that best identify the variance of the input distribution [22]. This technique is limited to linear representations, unlike the manifold organized by the autoencoder. Thus, the AE is able to construct higher fidelity correlations between the original and latent spaces, preserving relationships. Furthermore, there are multiple accounts in the literature in using the latent representation over the initial dataset with great effect for increased classification performance, better interpretability of the obtained clusters, or better ability to generalize over the latent representation [23–25].

Salakhutidnov and Hinton restrict the latent representation to a binary code which is interpreted as the output of a black-box hash function modelled by the encoder. The hashing is applied in the field of document retrieval, where the hashing of the query is used to retrieve the directly associated documents plus documents from neighbouring hashes. The task has also been approached from a generative approach by Hansen et al. in *Unsupervised Neural Generative Semantic Hashing* [26].

2.3. Reinforcement Learning. *Reinforcement learning* (RL) is a paradigm of the machine learning field where problems are modelled around two fundamental notions: agents and environments. Agents are able to interact with the environment via a defined set of "actions" which change the environment's "state." Defining the problem's solution as a desirable environment state, the agent is conditioned via "rewards" and "punishments" to reach this favourable state. RL purpose is to teach an agent the optimal policy of acting inside an environment. The environment can at any moment be in a certain *state* that can be changed by the agent's *actions*. The agent receives feedback from the environment in the form of a *reward*. Using a tradeoff between reward and value (future reward received by the agent by taking a certain action in a particular state), the agent learns a policy that decides the best course of action for a given state.

This flexible framework has allowed to model complex real-world situations: scheduling drug administration to patients with chronic conditions in order to minimise risk of negative interactions [27, 28], minimising energy costs associated with cooling of data centers [29], or outmatching human players in games such as Go [30].

RL is facing a growing interest in the discipline of algorithmic trading [31, 32]. The interest can be explained by the ease with which the problem can be modelled: given the price fluctuation of a certain instrument, an agent's purpose is to maximise the overall profit. Current frontier in reinforcement learning focuses on improved training performance, particularly incentivising the agent to explore multiple action courses and breaking the state causality effect on training by sampling and replaying random past states [33]. Intuitively, these improvements focus on offering the agent the ability to retrospect and decide on past better courses of action.

2.4. Time Series Generation. *GAN-based* methods or generative adversarial network models have emerged as a popular technique for generating or augmenting datasets, especially with images and videos. However, GANs give poor fidelity in networking data, which has both complex temporal correlations and mixed discrete-continuous data types. Although GAN-based time series generation exists—for instance for medical time series—such techniques fail on more complex data exhibiting poor autocorrelation scores on long sequences while prone to mode collapse.

TimeGAN architecture introduced by Yoon et al. [10] is of strong interest for the proposed method, as it reinforces the idea that latent spaces can be used to better understand the original time series distribution of the data. Specifically, the paper proposes using two latent spaces, H_S and H_X, where S represents the mathematical space of static features of the time series, e.g., gender, while X represents the space of temporal dependencies of the time series, e.g., the cholesterol level as the person ages. The paper asserts that

instead of using only a generator-discriminator system for creating new samples, introducing supervised learning to the unsupervised generation will increase the fidelity of generated data. The supervised loss comes from two encoder-decoder pairs (h: $S \longmapsto H_S, e$: $H_S \longmapsto S$) and (h_X: $X \longmapsto H_X$, e_X: $H_X \longmapsto X$) between the initial space and latent space, with the GAN networks learning to directly replicate the latent vectors: (g: $Z_S \longmapsto H_S, d$: $H_S \longmapsto$ IR) and (g_X: $Z_S \longmapsto H_S, d$: $H_S \longmapsto$ IR), where Z_S and Z_X are the space the random noise is sampled from. Of particular interest is the use of recurrent neural networks throughout the architecture. Notably, g_X features the use of a 2-step autoregression dependency for creating the temporal latent vector. Recurrent neural networks are also used for encoding and decoding between X and H_X.

DoppelGANger architecture introduced by Lin et al. [34] represents a current benchmark in time series generation. It tackles a similar problem with TimeGAN as both separate the generation of static attributes from the time series measurements, although focusing on privacy over accurate reproduction of the target distribution time series. Specifically, the generation procedure for the time series implies a conditional process akin to prior work with Conditional Generative Adversarial Nets [35].

The network further improves by providing a normalization approach that avoids mode collapse on long time series: instead of normalizing the entire data set at once, using the global minimum and maximum, the algorithm normalizes using the per-sample minimum and maximum. Furthermore, the minimum and maximum are attached as static metadata describing the associated time series. Thus, the generator is responsible for creating the static features and is also in charge of creating the features that normalize the time series.

The final DoppelGANger architecture uses three networks for generating data: a generator network used for generating static attributes and a generator network used for generating the minimum and maximum of each time series, as described above. The data generated by the two networks is fed into the third, a recurrent neural network which leverages the provided information plus its internal state to generate measurements. A stacked discriminator critiques the generator's work: one model focuses on the generated static features (also called metadata), while the other is a recurrent neural network critiquing the generated measurements.

3. Methodology

This section presents the steps undertaken in obtaining the augmented dataset. The dataset is discussed, along with the data preprocessing operations carried out. Afterwards, the *LSTM-based autoencoder architecture* employed for translating between the initial and the latent space is detailed upon. The latent space obtained is explored, and underrepresented sequences are identified using an *OPTICS clustering algorithm*. Moving on, the *adversarial network* (WGAN) used to sample new examples is presented, and the process of integrating the new samples into the initial dataset is discussed.

A high-level overview of the DAuGAN approach is depicted in Figure 1. The code, models, and dataset used are publicly available in [36].

3.1. Dataset. For the purpose of the paper, a dataset describing the evolution of the price for Apple's company stock, denoted by the AAPL ticker on New York stock exchange, has been chosen.

The dataset presents samples at every 15 minutes, covering the company's price evolution starting from 1^{st} of January 1998 until $3^{r\,d}$ of December 2021, totalling 289487 time steps. The dataset features 7 initial columns: date, time, open, high, low, close, and volume. These features are often used in the domain of algorithmic trading and offer indicators on the price's evolution per time step. The open feature describes the price at the start of the time step, and high and low describe the maximum and minimum reached throughout the time-step, while the close price describes the price at interval's end. The volume feature represents the number of trades executed in the given period.

Table 1 presents a sample fragment from the beginning of the time series.

Since the time steps are continuous, there are constraints that apply for any time step t.

$$\forall t \geq 1\colon \text{close}_{t-1} = \text{open}_t, \tag{1}$$

$$x_t \leq \text{high}_t, \forall t, \forall x \in \{\text{low}, \text{close}, \text{open}\}, \tag{2}$$

$$x_t \geq \text{close}_t, \forall t, \forall x \in \{\text{low}, \text{high}, \text{open}\}. \tag{3}$$

However, real world imposes exceptions to these constraints. First, the data used come only from the trading hours action, starting from 09 : 30 until 16 : 00, Monday to Friday, when all traders can take part in the market. However, the NYSE, and in general, the exchanges located in the United States, also present a "before-market" and "after-market" period, limited to institutional investors such as pension funds, hedge funds, or banks. Furthermore, shares can be traded between any two interested parties, without the exchange as an intermediary.

It is beyond the scope of the paper to identify and enumerate all possible sources of discontinuity that could violate Equation (1). The simplifying assumption that the condition holds for any two consecutive steps is made.

Basic statistics for the numerical features of the dataset are calculated. The analysis from Table 2 reveals that volume features a very wide, $[10^2, 10^7]$, domain.

Columns open and volume are plotted, observing that columns high, low, and close will trend in correlation and close to open column, leading to Figure 2. Figure 2(a) presents the histogram of volume column in the initial dataset, while Figure 2(b) depicts the open column evolution in time.

The keen observer will be very interested in the two sudden drops in price illustrated in open column evolution illustrated in Figure 2(b). They represent a domain-specific event called *share splitting*. In a X: 1 share split, the price of one share is divided by X while each share presplit is

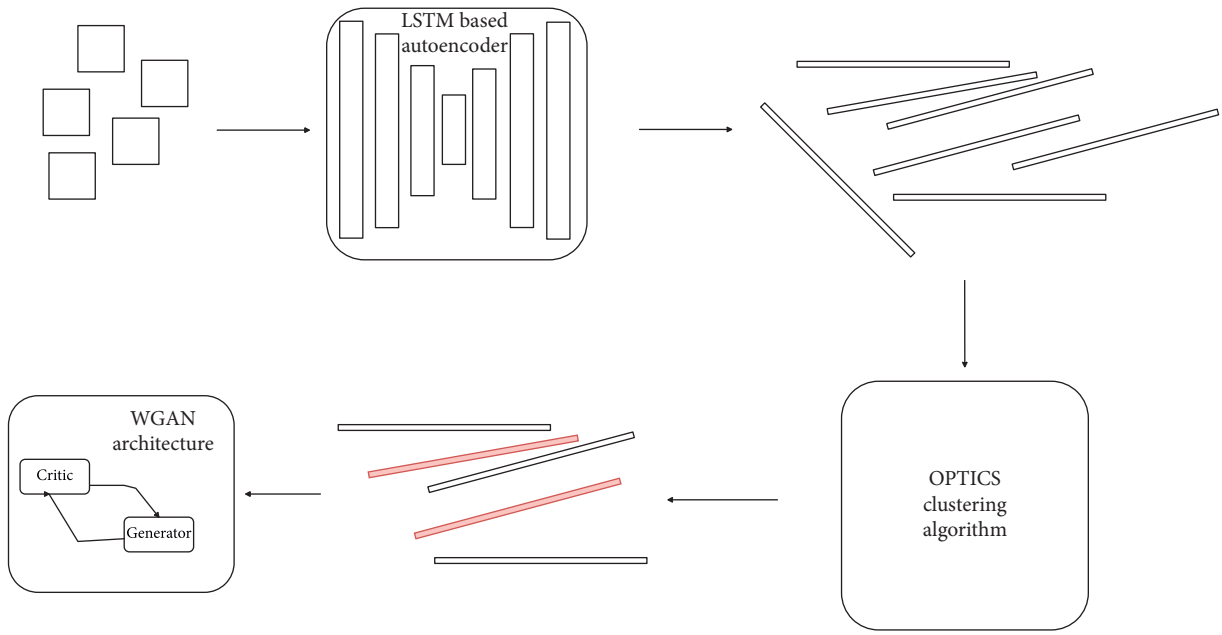

Figure 1: DAuGAN approach.

Table 1: An excerpt from the beginning of the time series.

Date	Time	Open	High	Low	Close	Volume
1998/02/01	09:30	13.6250	13.7500	13.5000	13.6875	20270
1998/02/01	09:45	13.6875	13.7500	13.5000	13.6250	334000
1998/02/01	10:00	13.6250	13.7500	13.5625	13.7500	299900
1998/02/01	10:15	13.7500	14.0000	13.6250	14.0000	430201
1998/02/01	10:30	13.9375	14.8125	13.7500	14.6250	944200
1998/02/01	10:45	14.6250	14.7500	14.3750	14.4375	218103

Table 2: Count, mean, standard deviation, and minimum values for the entire dataset plus upper bounds for each quarter of the dataset in sorted order.

	Open	High	Low	Close	Volume
Count	$2.894870e+05$	$2.894870e+05$	$2.894870e+05$	$2.894870e+05$	$2.894870e+05$
Mean	187.757776	188.038462	187.466230	187.758491	$4.580510e+05$
Std	160.514466	160.689723	160.326415	160.513598	$9.206810e+05$
Min	12.550000	12.950000	11.312500	12.850000	$1.000000e+02$
25%	78.750000	78.920000	78.500000	78.750000	$5.900000e+03$
50%	131.040000	131.330000	130.790000	131.040000	$1.033630e+05$
75%	248.160000	248.605000	247.625000	248.150000	$5.635505e+05$
Max	704.800000	705.070000	704.530000	704.800000	$7.514145e+07$

replaced with X times more. This preserves the value of the investment while lowering the financial bar for buying one share, attracting interest and activity from smaller investors with the better price per action.

3.1.1. Data Preprocessing. Date and time features are discarded, since the augmented dataset obtained at the end of the procedure contains a larger number of samples and the two features must be mocked. For training open, high, low, close, and volume columns are used. Considering the exponential distribution of the volume highlighted in Figure 2, with values in the domain of $[10^2, 10^7]$, a logarithm transformation is applied column-wise, followed two independent statistical normalization operations: one for the [open, high, low, close] columns and one for volume. Applying normalization on all columns would violate the constraint of Equation (1), as the features follow different distributions. Training on non-logarithmized leads to an autoencoder collapse, with most values outputted be decoder for volume being zero.

Figure 3 illustrates the dataset after normalization. One observes that the transformation of volume feature (Figure 3(a)) is notable, compared to the initial data (Figure 2(a)).

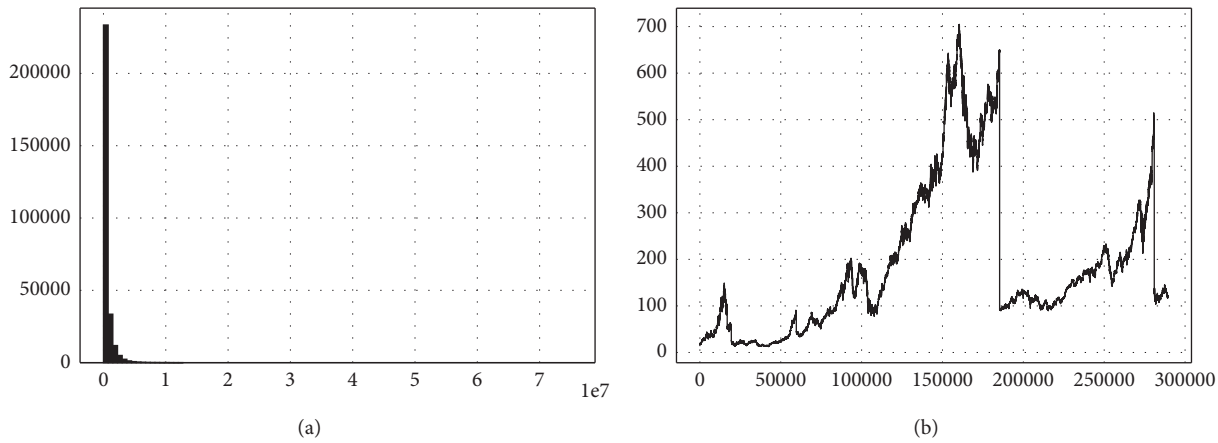

Figure 2: (a) Histogram of volume feature in the initial dataset; (b) open feature evolution in time.

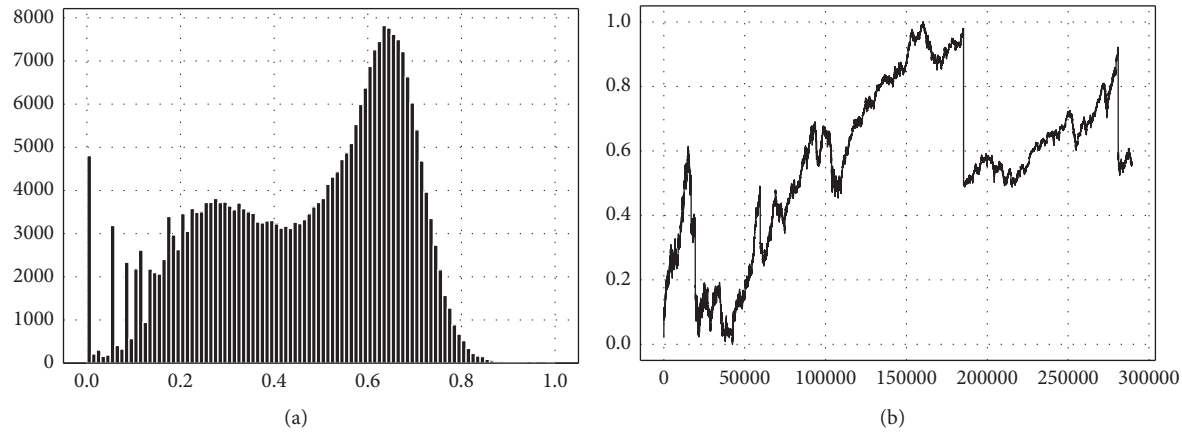

Figure 3: (a) Histogram of volume feature in the normalized dataset; (b) open feature evolution in time in the dataset after normalization.

In preparation for the training of the models, time series are split into chunks of twenty time steps and an overlap of eight time steps between two chunks formed of the forty consecutive time steps.

3.2. Autoencoder. It is difficult to identify outliers in the original, time series problem space. The issue is resolved via an autoencoder architecture that translates the time series samples into a latent, Euclidean space, where clustering can be applied in order to identify outliers. The autoencoder is tasked with translating between the two spaces. The dimensions of the latent space are not significant in themselves, and the cardinality has been chosen in order to minimise Pearson correlation, reducing autoencoder training time.

The architecture of the proposed autoencoder is illustrated in Table 3 (the architecture for the encoder) and Table 4 (the architecture for the decoder). The encoder and decoder parts of the architecture mirror each other. When reproducing the experiments, one should expect training and validation losses in the domain of 10^{-4} after 1000 epochs of training. Besides the val_{loss} metric, Person correlation between dimensions of latent space is employed in order to determine the minimum number of nonredundant dimensions.

In the encoder, one-dimensional convolutions use same padding mode, increasing the number of dimensions fed into the LSTM layers. It has been observed that the expansion in dimensionality aids the convergence of LSTM layers. Three stacked LSTM layers are used to capture temporal dependencies and encode them in the condensed latent form. The decoder operates in the same manner, with stacked LSTM layers expanding the temporal correlations encoded in latent and reverse convolutions condensing dimensions back to the original form. Following best practices from the literature, dropout layers with a rate $\beta = 0.3$ are intertwined for regularization purposes [37], and PReLu function is employed as activation between all layers [38]. The optimization algorithm involved in training is Stochastic Gradient Descent with a learning rate of $\alpha = 0.0001$ that uses Nesterov momentum [39]. The choice of the optimizer is motivated by the fact that Adam is not guaranteed to converge [40].

3.2.1. Analysing the Latent Space. The Pearson correlation heat map between the features is illustrated in Figure 4. The latent features have weak correlation, indicating an optimum number of features. An OPTICS algorithm [41] is applied for identifying minority clusters, sampling the epsilon

Table 3: Encoder architecture.

Layer (type)	Output shape	Param #
input_1 (InputLayer)	[(None; 20; 5)]	0
conv1d (Conv1D)	(None; 20; 256)	5376
p_re_lu (PReLU)	(None; 20; 256)	5120
dropout (Dropout)	(None; 20; 256)	0
conv1d_1 (Conv1D)	(None; 20; 256)	262400
p_re_lu_1 (PReLU)	(None; 20; 256)	5120
dropout_1 (Dropout)	(None; 20; 256)	0
conv1d_2 (Conv1D)	(None; 20; 256)	262400
p_re_lu_2 (PReLU)	(None; 20; 256)	5120
dropout_2 (Dropout)	(None; 20; 256)	0
lstm (LSTM)	(None; 20; 256)	525312
p_re_lu_3 (PReLU)	(None; 20; 256)	5120
dropout_3 (Dropout)	(None; 20; 256)	0
lstm_1 (LSTM)	(None; 20; 256)	525312
p_re_lu_4 (PReLU)	(None; 20; 256)	5120
dropout_4 (Dropout)	(None; 20; 256)	0
lstm_2 (LSTM)	(None; 12)	12912

Table 4: Decoder architecture.

Layer (type)	Output shape	Param #
input_2 (InputLayer)	[(None; 12)]	0
repeat_vector (RepeatVector)	(None; 20; 12)	0
lstm_3 (LSTM)	(None; 20; 256)	275456
p_re_lu_5 (PReLU)	(None; 20; 256)	5120
dropout_5 (Dropout)	(None; 20; 256)	0
lstm_4 (LSTM)	(None; 20; 256)	525312
p_re_lu_6 (PReLU)	(None; 20; 256)	5120
dropout_6 (Dropout)	(None; 20; 256)	0
conv1d_3 (Conv1D)	(None; 20; 256)	262400
p_re_lu_7 (PReLU)	(None; 20; 256)	5120
dropout_7 (Dropout)	(None; 20; 256)	0
conv1d_4 (Conv1D)	(None; 20; 256)	262400
p_re_lu_8 (PReLU)	(None; 20; 256)	5120
dropout_8 (Dropout)	(None; 20; 256)	0
conv1d_5 (Conv1D)	(None; 20; 256)	262400
p_re_lu_9 (PReLU)	(None; 20; 256)	5120
dropout_9 (Dropout)	(None; 20; 256)	0
time_distributed (TimeDistributed)	(None; 20; 5)	1285
time_distributed_1 (TimeDistributed)	(None; 20; 5)	0

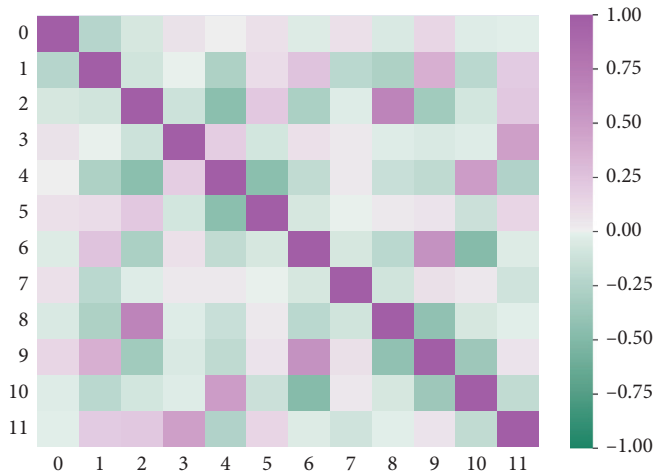

Figure 4: Pearson correlation between features.

hyperparameter linearly from the $[0.05, 0.5]$ range. The Elbow method [42] is applied on total variance in order to choose an optimum number of clusters. Analysis yields a majority cluster of $\approx$ 20000 points, with the rest being classified as outliers or noise. Principal component analysis [43] is employed for reducing dimensionality to three axes, thus allowing visualization.

3.3. Synthesising New Samples. The generative architecture presented in Tables 5 and 6 is able to synthesise credible examples that resemble the minority class. For statistical confirmation, a multivariate Wilcoxon test between the real minority points and the generated points is used [44, 45]. A p-value of 0.7275 is obtained at a significance level $\alpha = 0.05$, unable to refute the null hypothesis, proving there is no significant difference between the real and the generated points. The resemblance between distributions can be observed in Figure 5.

Figure 6 depicts the correlation heat map for the synthesised examples.

It is worth noting that the feature correlation for synthesised latent vectors slightly differs from the minority features' correlation, as shown in Figure 7.

3.3.1. Smoothing the Synthesised Examples. Despite statistical resemblance of the generated samples, the WGAN is unable to model the constraints from formulae (1), (2), and (3).

The issue is approached as an optimization problem: the closest point to the initially synthesised data point is found that satisfies all constraints simultaneously. Bayesian search [46] in tandem with a greedy algorithm is employed. The time steps corresponding to the latent point are iteratively "smoothed." If the time step's values do not respect the imposed constraints, the latent point will be assigned a negative value and a new neighbouring latent point will be verified. Should the time step be found adequate, it is assigned the inverse of the distance between the original location of the sampled synthetic point and the current location of the data point, and this metric is maximised. It should be noted that the original data point is kept as reference throughout all explorations.

Formally, let $\mathrm{d} = \|\mathrm{clv} - \mathrm{olv}\|^2$ (clv denotes the current latent vector, and olv represents the original latent vector); Bayesian search must optimize the black-box function γ for each time step, minimising the distance d at the overall data point level. After fixing a time step, the open price of the next time step is set to be equal to the close price of the current one in order to preserve continuity.

$$\gamma(\mathrm{open}, \mathrm{high}, \mathrm{low}, \mathrm{close}) = \begin{cases} -1, & \text{if (time step passes)}, \\ \dfrac{1}{d}, & \text{otherwise.} \end{cases} \tag{4}$$

The search space of the Bayesian process over γ is constrained to $X \pm \sqrt{X}$, where X is the value for any of the features. The point with the maximum score after a set number of steps is selected and integrated in the time series. For time steps with index $t \geq 1$, the search is carried out only for $\{\mathrm{high}, \mathrm{low}, \mathrm{close}\}$, since open price has been set at the first step.

In order to integrate a smoothed chunk ζ into the original time series, a tuple of consecutive chunks, defined by index $t < \mathrm{len}(\mathrm{time\_series}) - 1$, is identified such that the objective is minimised:

$$\left\| \left(\mathrm{close}_t, \mathrm{open}_{t+1}, \gamma\left(\mathrm{volume}_t \colon \mathrm{volume}_{t+1}\right) - \left(\mathrm{open}_\zeta, \mathrm{close}_\zeta, \gamma\left(\mathrm{volume}_\zeta\right)\right\|^2. \right. \tag{5}$$

Minimising the objective is equivalent to minimising the distance between $(\mathrm{close}_t, \mathrm{open}_\zeta)$ and $(\mathrm{close}_\zeta, \mathrm{open}_{t+1})$, preserving the overall smoothness of the time series. Table 7 presents a chunk fragment obtained in the generation process.

4. Results and Discussion

With the goal of answering research question RQ2, Section 4.1 presents the benchmark used to assess the effectiveness of the augmentation introduced in Section 3 in improving the performance of trading algorithms. The obtained results are then presented in Section 4.2.

4.1. Benchmark Methodology and Data Preparation. In order to assess the impact of augmentation, independent reinforcement learning trading algorithms and domain-specific data preparation procedures are provided by the open-source *FinRL* library [47].

For the benchmark, 80% of the original time series is kept for training the trading strategies, while employing the rest of 20% for blind validation. For fair comparison, the augmentation procedure is employed only on the training portion, and the impact is measured on the validation period. The algorithms are trained on two distinct datasets, the augmented training dataset and original training dataset, until convergence, and score on validation is measured.

FinRL preprocessing is applied on both datasets. The procedure adds technical indicators, whose purpose is to highlight trends in a security's price evolution, such as relative volatility, magnitude of price shifts, or trends in price shift. The indicators added by *FinRL* are 12-MACD, Bollinger Bands, 30-RelativeStrengthIndex, 30-CommodityChannelIndex, 30-AverageDirectionalIndex, 30-CloseSimpleMovingAverage, and 60-CloseSimpleMovAvg.

The *FinRL* library requires the presence of two extra columns in order to calculate the technical indicators: the tic column which represents the security's descriptor (*FinRL* is

Table 5: WGAN generator architecture.

Layer (type)	Output shape	Param #
input_4 (InputLayer)	[(None; 15; 1)]	0
dense_3 (Dense)	(None; 15; 20)	40
conv1d_9 (Conv1D)	(None; 8; 16)	1296
leaky_re_lu_3 (LeakyReLU)	(None; 8; 16)	0
conv1d_10 (Conv1D)	(None; 4; 16)	1040
leaky_re_lu_4 (LeakyReLU)	(None; 4; 16)	0
flatten_1 (Flatten)	(None; 64)	0
dense_4 (Dense)	(None; 100)	6500
dense_5 (Dense)	(None; 100)	10100
dense_6 (ense)	(None; 12)	1212

Table 6: WGAN critic architecture.

Layer (type)	Output shape	Param #
input_3 (InputLayer)	[(None; 12; 1)]	0
conv1d_6 (Conv1D)	(None; 6; 16)	80
leaky_re_lu (LeakyReLU)	(None; 6; 16)	0
conv1d_7 (Conv1D)	(None; 3; 16)	1040
leaky_re_lu_1 (LeakyReLU)	(None; 3; 16)	0
conv1d_8 (Conv1D)	(None; 2; 16)	1040
leaky_re_lu_2 (LeakyReLU)	(None; 2; 16)	0
flatten (Flatten)	(None; 32)	0
dense_1 (Dense)	(None; 100)	3300
dense_2 (Dense)	(None; 1)	101

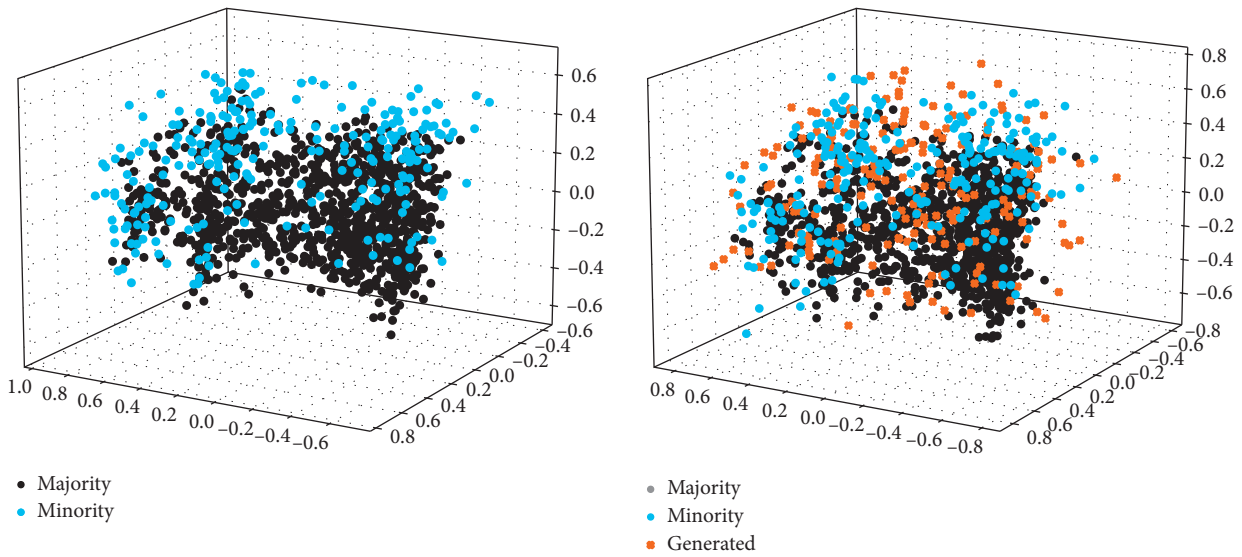

Figure 5: Distribution of latent points before and after synthesising new examples.

able to trade multiple securities at once, hence the requirement for this column) and the date column which represents the date and time of the column. With the simplifying assumption that the security has been traded continuously in intervals of 15 minutes starting from an arbitrary date, the two columns are added.

The trading agent's state is described by the tuple (shares, capital), where shares describes the number of owned shares, while capital describes the amount of monetary units available for buying shares. The state of the agent is completed by the time series, as seen until moment t of time. The initial state $(\text{shares}_0, \text{capital}_0) = (0, 200000)$. The value of a portfolio value with shares is defined as $\text{capital}_t + \sum_{0,r} \text{open}_t$.

At any time step, the agent's action space spans the integers $[-\min(k, \text{shares}_t), k]$. k is a positive integer hyperparameter set in the environment. Negative integers indicate selling that amount of shares, receiving an amount of capital equal to the opening price for each sold share.

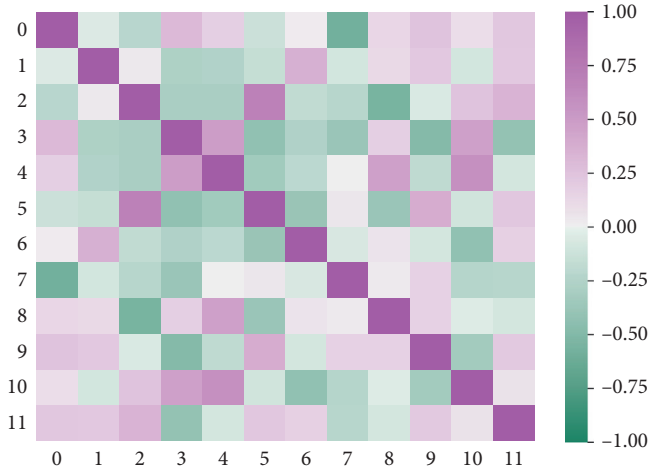

Figure 6: Correlation heat map for generated examples.

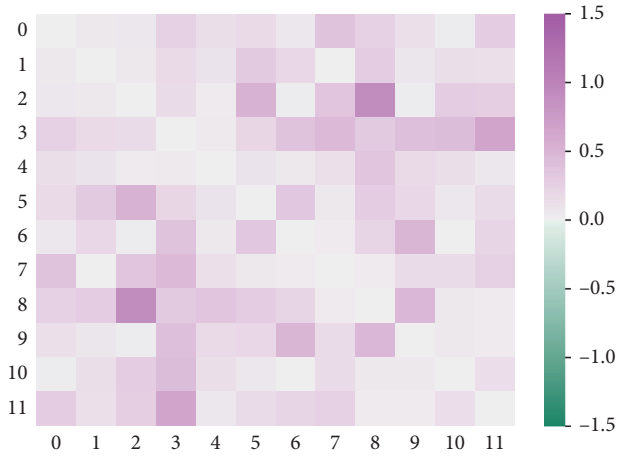

Figure 7: Absolute difference between the correlation maps of minority and generated examples.

Table 7: Generated chunk fragment.

	Open	High	Low	Close	Volume
0	326.366272	341.629822	314.827942	330.429718	517.638489
1	330.429718	336.826141	310.153687	325.654053	899.540161
2	325.654053	338.448029	311.383545	327.100647	2897.673340
3	327.100647	351.454468	313.967712	351.090942	3598.678955
4	351.090942	340.434845	313.185944	329.017059	4032.858887
5	329.017059	340.508057	313.215607	329.068115	5119.424316
6	329.068115	335.061981	308.940491	324.127441	153.940964
7	324.127441	331.650665	305.705658	320.777008	144.449570
8	320.777008	335.563629	308.537842	324.214722	4437.651367
9	324.214722	340.812073	313.569183	329.397736	3368.686523
10	329.397736	340.041931	312.984558	328.713165	2112.148193
11	328.713165	344.093750	312.746857	319.489136	575.971191
12	319.489136	339.126801	312.318512	327.886292	1199.597290
13	327.886292	344.057251	316.442230	332.473938	9648.517578
14	332.473938	346.419373	318.820251	334.856781	5601.142578
15	334.856781	345.538300	318.073364	334.029663	4347.550781

Table 8: Improvement in trading performance, measured with respect to the number of introduced samples and algorithm used.

Algorithm	# of introduced samples		
	2000	4000	6000
DDPG	+2.71%	+3.44%	**+4.25%**
PPO	+3.12%	+3.47%	**+3.82%**
A2C	+1.89%	+2.31%	**+2.42%**

Table 9: Hyperparameters for DDPG, PPO, and A2C algorithms.

Algorithm	Hyperparameter	Value
DDPG	critic_learning_rate	$1e-3$
	actor_learning_rate	$1e-3$
	l2_weight_regularization	$1e-6$
	gradient_clipping	None
	train_batch_size	256
	episodes_before_learn_start	1500
PPO	sgd_lr	$5e-5$
	epochs_per_train_batch	30
	ppo_parameter_clipping	0.3
	kl_divergence_target	0.01
	value_function_clip_parameter	10
	entropy_coeff	0.0
A2C	Adam_learning_rate	$1e-4$
	Adam_minibatch_size	32
	batch_size	12500
	gradient_steps	3000
	tau_mov_avg	0.01
	l2_weight_regularization	$1e-6$
	critic_learning_rate	$1e-4$
	actor_learning_rate	$1e-4$

Positive amounts indicate buying stock units and have the reverse effect on capital. The special case $k = 0$, denoting the agent's choice to hold its current position, should be noted.

4.2. Results. A positive correlation between the amount of samples introduced in the original time series, and the performance improvement of the algorithm is identified. The improvement is defined as the difference in portfolio value between training on the original time series and augmented series.

Table 8 summarizes the performance improvement of DAuGANa. Permutations of introduced samples and benchmark algorithm used: Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), and Advantage Actor Critic (A2C). Regarding the hyperparameter setting, the algorithms were trained using the values depicted in Table 9.

5. Conclusions and Future Work

The paper has introduced a novel augmentation method for identifying poorly represented sections of a time series, studied the synthesis of new data points and their integration into the time series, and assessed the performance improvement on a benchmark machine learning model.

Data synthetisation is a valid training augmentation in the area of algorithmic trading, which has the potential to be extended to other domains involving time series, due to the generality of the latent space approach. Of interest for the future are mission-critical tasks such as detecting rare medical conditions or weather now-casting, where performance improvement is vital.

As possible improvements, the authors hypothesise that the use of recurrent neural networks at the generation step [10], combined with the autonormalization trick using DoppelGANger discussed in [34], can result in longer synthesised time series, eliminating the need for interleaving generated samples back into the original time series. Instead, numerous independent "training episodes," several chunks in length, could be fed to the reinforcement learning agent, a method known to improve training performance [48].

Data Availability

The data used to support the findings of this study are available at the aforementioned repository [36].

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The first author acknowledges the financial support received from Babeș-Bolyai University through the special

scholarship for scientific activity for the academic year 2020–2021. The research leading to these results has received funding from the NO Grants 2014–2021, under Project contract no. 26/2020.

References

[1] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, 2019.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2011.

[3] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers, "Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks," *Scientific Reports*, vol. 9, no. 1, 2019.

[4] M. Mustafa, B. Deborah, B. Wahid, L. Zarija, A.-R. Rami, and M. K. Jan, "CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks," *Comput. Astrophys.*vol. 6, p. 1, 2019.

[5] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks,," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, IEEE Computer Society, Los Alamitos, CA, USA, june 2019.

[6] M. Pasini, "MelGAN-VC: voice conversion and audio style transfer on arbitrarily long samples using spectrograms," *Electrical Engineering and Systems Science*, https://arxiv.org/abs/1910.03713, 2019.

[7] M. Chauvet, "Stock market fluctuations and the business cycle," *SSRN Electronic Journal*, 2001.

[8] S. Edwards, J. G. Biscarri, and F. P. de Gracia, "Stock market cycles, financial liberalization and volatility," *Technical Report D*, 2003.

[9] N. Tavakoli, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Clustering time series data through autoencoder-based deep learning models," 2020, https://arxiv.org/abs/2004.07296.

[10] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," Edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., Vancover, Canada, December 2019.

[11] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," Edited by K. Chaudhuri and R. Salakhutdinov, Eds., in *Proceedings of the 36th International Conference on Machine Learning, Ser. Proceedings of Machine Learning Research*, vol. 97, pp. 7354–7363pp. 7354–, Long Beach, CA, USA, June 2019.

[12] I. J. Goodfellow, "NIPS 2016 tutorial: generative adversarial networks," 2017, http://arxiv.org/abs/1701.00160.

[13] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *Journal of Computational Chemistry*, vol. 38, no. 16, pp. 1291–1307, 2017.

[14] J. Kolen, *A Field Guide to Dynamical Recurrent Networks*, IEEE Press, New York, NY, USA, 2001.

[15] N. Kodali, J. D. Abernethy, J. Hays, and Z. Kira, "How to train your DRAGAN," 2017, http://arxiv.org/abs/1705.07215.

[16] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017, http://arxiv.org/abs/1703.10593.

[17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, IEEE, Venice, Italy, October 2017.

[18] M. Amodio, S. Krishnaswamy, and " Travelgan, "Image-to-image translation by transformation vector learning," 2019, http://arxiv.org/abs/1902.09631.

[19] M. Amodio and S. Krishnaswamy, ""TraVeLGAN: image-to-image translation by transformation vector learning," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8975–8984, IEEE, Long Beach, CA, USA, June 2019.

[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 214–223, Sydney, Australia, August 2017.

[21] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2020, https://arxiv.org/abs/2003.05991.

[22] K. Pearson and L. I. I. I. ", "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[23] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[24] I. Goodfellow, *Deep Learning*, The MIT Press, Cambridge, MA, USA, 2016.

[25] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[26] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma, "Unsupervised neural generative semantic hashing," http://arxiv.org/abs/1906.00671.

[27] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: a survey," 2019, http://arxiv.org/abs/1908.08796.

[28] L. Wang, W. Zhang, X. He, and H. Zha, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ser. KDD '18*, pp. 2447–2456, Association for Computing Machinery, New York, NY, USA, July 2018.

[29] Y. Li, Y. Wen, D. Tao, and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 2002–2013, 2020.

[30] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[31] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: an ensemble strategy," in *Proceedings of the ICAIF '20: ACM International Conference on AI in Finance*, Association for Computing Machinery, New York, NY, USA, October 2020.

[32] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *Expert Systems with Applications*, vol. 173, Article ID 114632, 2021.

[33] D. Mehta, "State-of-the-Art reinforcement learning algorithms," *International Journal of Engineering Research and Technology*, vol. 8, pp. 717–722, 2020.

[34] Z. Lin, A. Jain, C. Wang, G. C. Fanti, and V. Sekar, "Generating high-fidelity, synthetic time series datasets with doppelganger," http://arxiv.org/abs/1909.13403.

[35] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, http://arxiv.org/abs/1411.1784.
[36] A. Bratu, "Daugan paper code and dataset repository," 2021, https://github.com/andreibratu/bachelor/tree/master/thesis.
[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, and " Dropout, "A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," http://arxiv.org/abs/1502.01852.
[39] A. Botev, G. Lever, and D. Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent," in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017.
[40] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proceedings of the International Conference on Learning Representations*, Vanconer, Canada, May 2018.
[41] M. Ankerst, M. M. Breunig, H. peter Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *Proceedings of the . ACM SIGMOD'99 Int. Conf. on Management of Data*, pp. 49–60, ACM Press, Philadelphia, PA, USA, June 1999.
[42] M. A. Syakur, B. K. Khotimah, E. M. S. Rochman, and B. D. Satoto, "Integration k-means clustering method and elbow method for identification of the best customer profile cluster," *IOP Conference Series: Materials Science and Engineering*, vol. 336, Article ID 012017, 2018.
[43] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, vol. 374, no. 2065, Article ID 20150202, 2016.
[44] H. Oja and R. H. Randles, "Multivariate nonparametric tests," *Statistical Science*, vol. 19, no. 4, 2004.
[45] C. fan Sheu and S. O'Curry, "Implementation of nonparametric multivariate statistics with s," *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 315–318, 1996.
[46] R. Turner, D. Eriksson, M. McCourt et al., "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: analysis of the black-box optimization challenge 2020," 2021, https://arxiv.org/abs/2104.10201.
[47] X.-Y. Liu, H. Yang, Q. Chen et al., "FinRL: a deep reinforcement learning library for automated stock trading in quantitative finance," *SSRN Electronic Journal*, https://arxiv.org/abs/2011.09607, 2020.
[48] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, http://arxiv.org/abs/1312.5602.