



پانزدهمین سمینار احتمال
و فرآیندهای تصادفی
۸ و ۹ شهریور ۱۴۰۴
دانشگاه کردستان



**Seminar On Probability
and Stochastic Processes**
August 30-31, 2025
University of Kurdistan



Optimizing Generative Adversarial Networks for Financial Time Series: A Stochastic Approach with Novel Loss Metrics

Pegah Moradi^{*1}, Seyed-Mohammad-Mahdi Kazemi²

¹Department of Financial Sciences, Kharazmi University, Tehran, Iran

²Department of Financial Sciences, Kharazmi University, Tehran, Iran

Abstract

Generative models for financial time series often struggle to model practical financial objectives, which limits their applicability. In this study, we propose an improved Generative Adversarial Network framework that combines advanced sequence modeling structures, such as Temporal Convolutional Networks capable of capturing long-range dependencies and volatility patterns, with a novel generator function. This generator function is explicitly constructed to ensure that the resulting stochastic process can transition to its corresponding risk-neutral distribution. It also incorporates a new type of loss function, designed based on incorporating financial performance measures. These components aim to improve the profitability of trading strategies, enhance risk-adjusted returns, and manage the volatility of financial outcomes. Our numerical results demonstrate that the distributional characteristics at both short and long time lags closely match the expected patterns, indicating that the methodology can lead to the generation of more realistic and actionable market scenarios for robust financial decision-making.

Keywords: Generative Adversarial Networks, Temporal Convolutional Networks, Stochastic Process, Time Series

Mathematics Subject Classification (2020): 62M10, 91B84, 65Cxx.

1 Introduction

Time series forecasting has been a fundamental area of research for many years. Most real-world processes naturally follow temporal structures, particularly in finance and economics, where variables such as asset prices, inflation rates, and other macroeconomic indicators evolve over time. In this study, our goal is to move beyond traditional forecasting methods by using Generative Adversarial Networks (GANs) to predict financial time series. In the financial domain, GANs present a powerful framework for modeling the complex dynamics of market

^{*}Corresponding author, pegah.moradi@khu.ac.ir

data. They can be employed to evaluate trading strategies. Their ability to learn and replicate sophisticated market behaviors makes GANs a promising tool for financial decision-making. Here we consider the problem of approximating a realistic asset price simulator by proposing a neural-network-based simulator trained using GAN techniques. This approach positions GANs between purely data-based approaches, such as historical simulations and traditional model-driven methods like Monte Carlo simulations that assume an underlying stock price model like the Black-Scholes model, Heston's stochastic volatility model, or Lévy processes. Unlike these classical methods, GANs offer flexibility without relying on strict assumptions about the underlying price dynamics.

However, applying GANs to sequential data requires models that can effectively capture temporal dependencies. Architectures such as Temporal Convolutional Networks (TCNs) are often integrated within the GAN frameworks to capture these sequential patterns. This specialized architecture allows the generator to produce sequences that respect the inherent order and memory present in time series data, which is essential for financial applications, where past behavior often influences future outcomes.

2 Temporal Convolutional Networks

TCNs are deep learning architectures that have shown strong performance on sequence data and was introduced by Bai and *et al.* (2018). TCNs are built using the concept of dilated causal convolutions, where each output at time t depends only on inputs at time t and earlier. Dilated convolutions allow networks to cover larger contexts without increasing computation. From now on, we assume $N_I, N_O, K, D, T \in \mathbb{N}$, which represent the input and output dimensions, kernel size, dilation factor and the length.

Definition 2.1. Let $X \in \mathbb{R}^{N_I \times T}$ be an N_I -variate sequence of length T and $W \in \mathbb{R}^{K \times N_I \times N_O}$ a tensor. For each time step $t \in \{D(K-1)+1, \dots, T\}$ and $m \in \{1, \dots, N_O\}$, We define $*_D$ Operator, which is a dilated causal convolution with dilation D and kernel size K as:

$$(W *_D X)_{m,t} := \sum_{i=1}^K \sum_{j=1}^{N_I} W_{i,j,m} \cdot X_{j,t-D(K-i)}.$$

TCNs are composed by combining dilated causal convolutions with nonlinearities. To improve clarity, a modular design called block module is used. A function $\psi : \mathbb{R}^{N_I \times T} \rightarrow \mathbb{R}^{N_O \times (T-S)}$ that is Lipschitz continuous is a block module with parameters (N_I, N_O, S) .

Definition 2.2. Let $T_0, L, N_0, \dots, N_{L+1} \in \mathbb{N}$, and for each $l \in \{1, \dots, L\}$, let $S_l \in \mathbb{N}$ such that $\sum_{l=1}^L S_l \leq T_0 - 1$. For $T_l := T_{l-1} - S_l$ it holds

$$T_L = T_0 - \sum_{l=1}^L S_l \geq 1.$$

Additionally, Let $\psi_l : \mathbb{R}^{N_{l-1} \times T_{l-1}} \rightarrow \mathbb{R}^{N_l \times T_l}$ be block modules, and let $w : \mathbb{R}^{N_L \times T_L} \rightarrow \mathbb{R}^{N_{L+1} \times T_L}$ be a 1×1 convolutional layer. The network is defined by the function $f : \mathbb{R}^{N_0 \times T_0} \times \Theta \rightarrow \mathbb{R}^{N_{L+1} \times T_L}$ as follows

$$f(X, \theta) = w \circ \psi_L \circ \dots \circ \psi_1(X).$$

This is a TCN with L hidden layers. The class of such networks from $\mathbb{R}^{d_0}$ to $\mathbb{R}^{d_1}$ is denoted $TCN_{d_0, d_1, L}$, where $d_0 = N_0$ and $d_1 = N_{L+1}$.

Definition 2.3. Let $f \in TCN_{N_0, N_{L+1}, L}$ be such that each ψ_l is composed of a causal convolution w_l with parameters $(N_{l-1}, N_l, K-l, D_l)$ and an activation function ϕ , i.e. $\psi_l = \phi \circ w_l$. Then $f : \mathbb{R}^{N_0 \times T_0} \times \Theta \rightarrow \mathbb{R}^{N_{L+1} \times T_L}$ is called a Vanilla TCN. If $D_l = D_{l-1}$ for all l , then the dilation factor is said to have a uniform dilation factor D . Similarly, if $K_l = K$ for all l , it is said to have a fixed kernel size K .

The ability of TCNs to model long-range temporal dependencies is evident when comparing their output to other models. We call the number of sequence elements that the TCN can capture the receptive field size [Heaton and Goodfellow \(2016\)](#).

Definition 2.4. Let $f \in \text{TCN}_{d_0, d_1, L}$, and let the sequence of modules $S_1, \dots, S_L$ be defined as in [Definition 2.2](#). The value below is referred to as the receptive field size (RFS).

$$T^{(f)} := 1 + \sum_{l=1}^L S_l$$

Furthermore, skip connections can also be incorporated into TCN architectures to improve representational flexibility.

Definition 2.5. Assume the same notation as in [Definition 2.2](#), and let $N_{\text{skip}} \in \mathbb{N}$.

$$\gamma_l : \mathbb{R}^{N_{l-1} \times T_{l-1}} \rightarrow \mathbb{R}^{N_l \times T_l} \times \mathbb{R}^{N_{\text{skip}} \times T_L} \quad \text{for } l \in \{1, \dots, L\}$$

Let γ be a block module with parameters $(N_{\text{skip}}, N_{L+1}, 0)$. If the output $Y \in \mathbb{R}^{N_{L+1} \times T_L}$ of the TCN function $f : \mathbb{R}^{N_0 \times T_0} \times \Theta \rightarrow \mathbb{R}^{N_{L+1} \times T_L}$ is defined recursively as

$$\begin{aligned} (X^{(l)}, H^{(l)}) &= \gamma_l(X^{(l-1)}) \quad \text{for } l \in \{1, \dots, L\} \\ Y &= \gamma\left(\sum_{l=1}^L H^{(l)}\right) \end{aligned}$$

with initial input $X^{(0)} \in \mathbb{R}^{N_0 \times T_0}$, then f is referred to as a TCN with skip connections.

One limitation of TCNs is that the sequence length they can process is constrained by the RFS. Capturing long-term temporal dependencies requires a significantly large RFS, which can lead to computational demands.

3 Generative Adversarial Networks

GAN models, introduced by [Goodfellow and et al. \(2014\)](#) have become an important concept in deep learning, as they can generate synthetic data that closely follows the characteristics of a given dataset. At their core, GANs employ a competitive framework between two neural networks: a generator and a discriminator. The generator strives to create data samples, that are indistinguishable from real data. Simultaneously, the discriminator tries to accurately differentiate between the real and the synthetic samples produced by the generator. This section begins by outlining the GAN framework for random variables and then expands the discussion to discrete-time stochastic processes, both employed by [Wiese and et al. \(2020\)](#).

3.1 GANs for Random Variables

In this context, assume that $(\mathbb{R}^{N_Z}, \mathcal{B}(\mathbb{R}^{N_Z}))$ and $(\mathbb{R}^{N_X}, \mathcal{B}(\mathbb{R}^{N_X}))$ represent the latent and data measure spaces, respectively. The latent variable Z acts as the input noise (prior), while X represents the target (data) variable. The main purpose of the generator $g : \mathbb{R}^{N_Z} \times \Theta(g) \rightarrow \mathbb{R}^{N_X}$ is to transform the noise into synthetic samples such that the random variable defined by $g_\theta(Z) := g_\theta \odot Z$, for some $\theta \in \Theta(g)$, shares the same distribution as X , denoted by $g_\theta(Z) \stackrel{d}{=} X$.

Definition 3.1. Let $g : \mathbb{R}^{N_Z} \times \Theta(g) \rightarrow \mathbb{R}^{N_X}$ be a neural network parameterized by $\theta \in \Theta(g)$.

$$\tilde{X} : \Omega \times \Theta(g) \rightarrow \mathbb{R}^{N_X}, \quad (\omega, \theta) \mapsto g_\theta(Z(\omega)).$$

Where $\tilde{X}$ is the generated random variable, the function g is the generator, and $\tilde{X}_\theta$ is the synthetic data generated using parameters θ .

Definition 3.2. Let $\tilde{d} : \mathbb{R}^{N_X} \times \Theta(d) \rightarrow \mathbb{R}$ be a neural network with parameter space $\Theta(d)$ and let $\sigma(x)$ be the activation function. We define the discriminator $d : \mathbb{R}^{N_X} \times \Theta(d) \rightarrow [0, 1]$ by

$$d(x, \eta) := \sigma \odot (\tilde{d}_\eta(x)).$$

The GAN training procedure can be viewed as a two players in a zero-sum game. The generator's objective is to produce data points $\{\tilde{x}_{\theta,i}\}_{i=1}^M$ that appear indistinguishable from real samples to the discriminator. Conversely, the discriminator's role is to classify whether a given input comes from the true data distribution or from the generator. The learning procedure is defined in two parts. the discriminator parameters $\eta \in \Theta(d)$ are optimized to maximize the following objective function for a fixed generator parameter $\theta \in \Theta(g)$:

$$L(\theta, \eta) := \mathbb{E}[\log d_\eta(X)] + \mathbb{E}[\log(1 - d_\eta(g_\theta(Z)))] = \mathbb{E}[\log d_\eta(X)] + \mathbb{E}[\log(1 - d_\eta(\tilde{X}\theta))].$$

This function encourages the discriminator to assign high scores to real data and low scores to generated data. Subsequently, the generator parameters θ are updated to minimize the probability that its outputs are correctly identified as synthetic. The overall training is structured as below, which is known as the standard GAN training objective.

$$\min_{\theta \in \Theta(g)} \max_{\eta \in \Theta(d)} L(\theta, \eta),$$

3.2 GANs for Stochastic Processes

To generalize GANs for modeling sequences or time series, we consider a stochastic process $(X_t)_{t \in \mathbb{Z}}$ parameterized by $\theta \in \Theta$. For $s, t \in \mathbb{Z}$ with $s \leq t$, we define:

$$X_{s:t,\theta} := (X_{s,\theta}, \dots, X_{t,\theta}), \quad X_{s:t,\theta}(\omega) := (X_{s,\theta}(\omega), \dots, X_{t,\theta}(\omega)) \in \mathbb{R}^{N_X \times (t-s+1)}.$$

Definition 3.3. Let $(Z_t)_{t \in \mathbb{Z}}$ be a sequence of i.i.d. noise process with values in $\mathbb{R}^{N_Z}$. Let

$$g : \mathbb{R}^{N_Z \times T^{(g)}} \times \Theta(g) \rightarrow \mathbb{R}^{N_X}$$

be a TCN with receptive field size $T^{(g)}$. Define the neural process $\tilde{X}$ as:

$$\tilde{X} : \Omega \times \mathbb{Z} \times \Theta(g) \rightarrow \mathbb{R}^{N_X}, \quad (\omega, t, \theta) \mapsto g_\theta(Z_{t-(T^{(g)}-1):t}(\omega)).$$

Then the sequence $\tilde{X}_\theta := (\tilde{X}_{t,\theta})_{t \in \mathbb{Z}}$ is called a neural stochastic process, with each $\tilde{X}_{t,\theta}$ being F - $\mathbb{B}(\mathbb{R}^{N_X})$ -measurable.

In our framework, the noise input $Z = (Z_t)_{t \in \mathbb{Z}}$ follows a multivariate standard normal distribution, i.e., $Z_t \sim \mathcal{N}(0, I)$. The generator maps this noise sequence into a synthetic process $\tilde{X}_\theta$ using a TCN. The discriminator is also defined using a TCN:

$$d : \mathbb{R}^{N_X \times T^{(d)}} \times \Theta(d) \rightarrow [0, 1],$$

where $T^{(g)}$ is the receptive field size and the objective in this temporal setting becomes:

$$\min_{\theta \in \Theta(g)} \max_{\eta \in \Theta(d)} L(\theta, \eta),$$

where

$$L(\theta, \eta) := \mathbb{E}[\log d_\eta(X_{1:T^{(d)}})] + \mathbb{E}[\log(1 - d_\eta(\tilde{X}_{1:T^{(d)}}\theta))].$$

Training follows the same structure as in the random variable case. Samples from a generated neural process $\{\tilde{x}_{1:T^{(d)},\theta}^{(i)}\}_{i=1}^M$ and of the target distribution $\{x_{1:T^{(d)}}^{(i)}\}_{i=1}^M$ are passed through the discriminator, which outputs classification probabilities.

4 Stochastic Volatility Neural Networks

The generator architecture in our framework is constructed by the Stochastic Volatility Neural Network (SVNN). The SVNN consists of volatility, drift, and another network which models the innovations. This structure implies that the volatility and drift are generated based on historical latent variable processed through a shared TCN, whereas the innovation is generated

via a separate network. In other words, if we consider $Z = (Z_t)_{t \in \mathbb{Z}}$ be a sequence of i.i.d. Gaussian random variables taking values in $\mathbb{R}^{N_Z}$. Let a TCN as $g^{(\text{TCN})} : \mathbb{R}^{N_Z \times T^{(g)}} \times \Theta^{(\text{TCN})} \rightarrow \mathbb{R}^{2N_X}$ with RFS $T^{(g)}$, and another network $g^{(\varepsilon)} : \mathbb{R}^{N_Z} \times \Theta^{(\varepsilon)} \rightarrow \mathbb{R}^{N_X}$. Suppose that $\alpha \in \Theta^{(\text{TCN})}$ and $\beta \in \Theta^{(\varepsilon)}$ denote parameters of the networks. We define a stochastic process R as:

$$R : \Omega \times Z \times \Theta^{(\text{TCN})} \times \Theta^{(\varepsilon)} \rightarrow \mathbb{R}^{N_X},$$

$$(\omega, t, \alpha, \beta) \mapsto [\sigma_{t,\alpha} \odot \varepsilon_{t,\beta} + \mu_{t,\alpha}](\omega),$$

Where the components are defined as:

$$h_t := g_\alpha^{(\text{TCN})}(Z),$$

$$\sigma_{t,\alpha} := |h_{t,1:N_X}|,$$

$$\mu_{t,\alpha} := h_{t,(N_X+1):2N_X},$$

$$\varepsilon_{t,\beta} := g_\beta^{(\varepsilon)}(Z_t).$$

The process R induced by the SVNN is referred to as a log return neural process. Within this framework, the neural processes $\sigma_\alpha := (\sigma_{t,\alpha})_{t \in \mathbb{Z}}$, $\mu_\alpha := (\mu_{t,\alpha})_{t \in \mathbb{Z}}$, $\varepsilon_\beta := (\varepsilon_{t,\beta})_{t \in \mathbb{Z}}$ are respectively called the volatility, drift, and innovation neural processes.

5 Evaluating Metrics and Scoring Criteria

To analyze how well the models perform, we employ a selection of performance metrics and evaluation scores.

Earth Mover Distance (EMD): The EMD serves as a measure of how different two probability distributions are. Let $\mathbb{P}^h$ be the historical distribution and $\mathbb{P}^g$ be the generated distribution of log returns, the EMD is defined as

$$EMD(\mathbb{P}^h, \mathbb{P}^g) = \inf_{\pi \in \Pi(\mathbb{P}^h, \mathbb{P}^g)} \mathbb{E}_{(X,Y) \sim \pi}[|X - Y|]$$

where $\Pi(\mathbb{P}^h, \mathbb{P}^g)$ is the collection of all possible joint probability distributions and marginal distributions are $\mathbb{P}^h$ and $\mathbb{P}^g$.

DY Metric: This metric captures the divergence between the logarithmic probabilities of the historical and generated data across discretized intervals. For a given time lag $t \in \mathbb{N}$, the DY metric is defined as

$$DY(t) = \sum_x |\log P_t^h(A_{t,x}) - \log P_t^g(A_{t,x})|$$

Where P_t^h and P_t^g represent the probability density functions of the historical and generated t -differenced return trajectories for historical and generated data. The term $(A_{t,x})_x$ refers to a partitioning of the real line such that for a fixed t and all x , the approximate condition $\log P_t^h(A_{t,x}) = 5/T$ holds true, with T being the total number of historical log returns.

Autocorrelation Function (ACF) Score: This score evaluates how well the autocorrelation properties of generated time series align with those of real data. Let $r_{1:T}$ be the historical log return series, and let $\{r_{1:\tilde{T},\theta}^{(1)}, \dots, r_{1:\tilde{T},\theta}^{(M)}\}$ be the generated log return series of length $\tilde{T} \in \mathbb{N}$. The autocorrelation, is defined as a function of the time lag τ and a series $r_{1:T}$.

$$\mathcal{C}(\tau; r) = \text{Corr}(r_{t+\tau}, r_t)$$

By defining $C : \mathbb{R}^T \rightarrow [-1, 1]^S : r_{1:T} \mapsto (\mathcal{C}(1; r), \dots, \mathcal{C}(S; r))$ as the autocorrelation function up to a lag $S \leq T - 1$, the ACF(f) score is calculated for a function $f : \mathbb{R} \rightarrow \mathbb{R}$ as:

$$ACF(f) := \|C(f(r_{L:T})) - \frac{1}{M} \sum_{i=1}^M C(f(r_{1:T,\theta}^{(i)}))\|_2$$

The ACF score is computed using constants $S = 250$, $M = 500$, and $\tilde{T} = 4000$ for $f(x) = x$, $f(x) = x^2$, and $f(x) = |x|$, which are serial, squared, and absolute autocorrelation, respectively.

Leverage Effect Score: This metric is quantified by examining the correlation between lagged, squared log returns and the log returns themselves. This is expressed as:

$$\mathcal{L}(\tau; r) = \text{Corr}(r_{t+\tau}^2, r_t)$$

for a given lag τ . If $L : \mathbb{R}^T \rightarrow [-1, 1]^S : r_{1:T} \mapsto (\mathcal{L}(1; r), \dots, \mathcal{L}(S; r))$ denotes the leverage effect function up to lag $S \leq T - 1$, then the leverage effect score is defined by:

$$\|L(r_{1:T}) - \frac{1}{M} \sum_{i=1}^M L(r_{1:\tilde{T}, \theta}^{(i)})\|_2$$

5.1 Novel Loss Function for the Generator

To better align the GAN's generative process with financial objectives, a novel economics-driven loss function for the generator (L^G) is introduced by Vuletić and *et al.* (2024). This loss function augments the standard adversarial loss component with terms directly related to financial performance metrics. The general form of this loss function is:

$$L^G(x, \hat{x}) = J^{(G)}(x, \hat{x}) - \alpha \text{PnL}^*(x, \hat{x}) + \beta \text{MSE}(x, \hat{x}) - \gamma \text{SR}^*(x, \hat{x}) + \delta \text{STD}(x, \hat{x}) \quad (1)$$

Where x and $\hat{x}$ represents the true and generated values, $J^{(G)}$ is binary cross-entropy, which is the standard generator loss function. $\alpha, \beta, \gamma, \delta$ are non-negative hyperparameters that weigh the contribution of each financial term. The additional financial terms in (1) are defined as follows:

Profit and Loss Term (PnL*): This term aims to maximize the profitability of a simulated trading strategy based on the generator's forecasts.

$$\text{PnL}^*(x, \hat{x}) = \frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \text{PnL}_a^*(x_i, \hat{x}_i)$$

where $\text{PnL}_a^*(x_i, \hat{x}_i) = \tanh(k_{tanh} \hat{x}_i) x_i$. The hyperparameter k_{tanh} controls the accuracy of our approximation.

Mean Squared Error Term (MSE): This term encourages the generated values ($\hat{x}_i$) to be close to the actual realized values (x_i). The generator aims to minimize this term.

$$\text{MSE}(x, \hat{x}) = \frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} (x_i - \hat{x}_i)^2$$

Sharpe Ratio Term (SR*): This term encourages the generator to produce forecasts that would lead to a high risk-adjusted return, the generator aims to maximize this term.

$$\text{SR}^*(x, \hat{x}) = \frac{\frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \text{PnL}_a^*(x_i, \hat{x}_i)}{\sqrt{\frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} (\text{PnL}_a^*(x_i, \hat{x}_i) - \frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \text{PnL}_a^*(x, \hat{x}))^2}}$$

Standard Deviation of PnL Term (STD): This term, which the generator aims to minimize, penalizes high volatility in the PnL series and provide more stable performance.

$$\text{STD}(x, \hat{x}) = \sqrt{\frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} (\text{PnL}_a^*(x_i, \hat{x}_i) - \frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \text{PnL}_a^*(x, \hat{x}))^2}$$

By incorporating these financial terms, the generator is not only trained to fool the discriminator, but also to produce synthetic financial time series that exhibit desirable characteristics from a trading or investment perspective.

6 Methodology

We explore four distinct model structures: a pure TCN, a hybrid Fin-TCN, a constrained SVNN and for baseline comparison, a conventional GARCH model.

TCN: To evaluate the effectiveness of a pure TCN-based architecture, we apply the TCN directly to the log return series. The model utilizes a generator function with a RFS of $T^{(g)} = 127$, and the return process is described by

$$R_{t,\theta} = g_\theta(Z_{t-(T^{(g)}-1):t}), Z_t \stackrel{iid}{\sim} \mathcal{N}(0, I)$$

Fin-TCN: The architecture of this model is identical to the TCN. However, the generator's loss function is augmented with specific financial metrics, argued in Section 5.1.

Constrained SVNN: We utilize a constrained SVNN as described in Section 4. The return series is modeled as

$$R_{t,\theta} = \sigma_{t,\theta} \varepsilon_{t,\theta} + \mu_{t,\theta}$$

Here, volatility $\sigma_{t,\theta}$, drift $\mu_{t,\theta}$, and innovations $\varepsilon_{t,\theta}$ are constrained to follow an i.i.d. $\mathcal{N}(0, 1)$ distribution. The latent noise process again follows an i.i.d. Gaussian distribution $Z_t \stackrel{iid}{\sim} \mathcal{N}(0, I)$ with $N_Z = 3$. The innovation term is specified by $\varepsilon_{t,\theta} = Z_{t,1}$ for all $t \in \mathbb{Z}$.

GARCH(1,1) with Constant Drift: As a classical benchmark, we also consider a GARCH(1,1) model with constant drift, with parameters $\mu \in \mathbb{R}$, $\omega > 0$, $\alpha, \beta \in [0, 1]$ and the constraint $\alpha + \beta < 1$. The parameter vector is defined as $\theta = (\omega, \alpha, \beta, \mu)$.

$$\begin{aligned} R_{t,\theta} &= \xi_t + \mu \\ \xi_t &= \sigma_t \varepsilon_t \\ \sigma_t^2 &= \omega + \alpha \xi_{t-1}^2 + \beta \sigma_{t-1}^2 \\ \varepsilon_t &\stackrel{iid}{\sim} \mathcal{N}(0, 1) \end{aligned}$$

Discussion and Results

In this study, we employed historical price data for the ticker "Vabesader" (representing Bank Saderat Iran), an actively traded stock on the Tehran Stock Exchange. The dataset consists of log return values from June 8, 2009, to May 6, 2025. We first examine the summarized performance of the models under consideration: C-SVNN, Fin-TCN, TCN, and GARCH(1,1). The statistics of interest are the annualized Sharpe Ratio, mean daily PnL, MAE, and RMSE. The summary statistics are shown in table 1. We remark that the highest PnL and Sharpe Ratios are achieved by Fin-TCN, followed by TCN. We further note that GARCH(1,1) attains the lowest MAE. However, the lowest RMSE is also achieved by Fin-TCN.

Table 1: Summary of financial performance metrics over the models

Model	MAE	RMSE	PnL	SR
C-SVNN	0.0322	0.0373	-0.1333	-1.6934
Fin-TCN	0.0311	0.0356	0.1058	0.8841
TCN	0.0312	0.0357	-0.0655	-0.7065
GARCH(1,1)	0.0199	0.0454	-0.3092	-0.7354

Furthermore, Table 2 presents the values of the evaluated metrics and scoring criteria for each of the models. Our numerical results highlight that C-SVNN model can learn a neural

process that matches the empirical distribution and dependence properties better than the benchmarked models. Specifically, it outperforms the GARCH(1,1) model in replicating both the empirical distribution and the temporal dependence structure of the observed returns.

Table 2: Evaluated metrics for the three models applied

Metric	C-SVNN	Fin-TCN	TCN	GARCH(1,1)
EMD(1)	0.0079	0.0136	0.0126	0.0134
EMD(5)	0.0208	0.0427	0.0399	0.0423
EMD(20)	0.0600	0.1036	0.0987	0.1031
EMD(100)	0.0879	0.2593	0.2541	0.2587
DY(1)	0.1179	12.1894	6.5783	8.4348
DY(5)	0.0719	16.4804	11.7203	12.4322
DY(20)	0.1522	16.3786	11.5897	14.0046
DY(100)	1.1441	18.3549	16.4555	18.3813
ACF(id)	0.1567	0.4906	0.3717	0.1641
ACF(abs)	0.5061	0.5379	0.5147	0.4415
ACF(sq)	0.038	0.0663	0.1118	0.3146
Leverage_Effect	0.7394	1.2436	0.8513	2.723804

This study introduces a novel loss function based on financial metrics like PnL and Sharpe Ratio, integrated with advanced sequence models such as SVNNs and TCNs, to improve the generation of synthetic financial time series. Unlike traditional GANs that often produce data lacking financial relevance despite statistical realism, this method directs the model to generate sequences that align with real-world financial performance criteria. Overall, this approach bridges the gap between statistical accuracy and financial applicability, providing a practical method for generating synthetic market data. It underscores the importance of aligning generative modeling techniques with the unique demands of the financial domain, providing a foundation for more effective applications in quantitative finance.

References

- Bai S, Kolter JZ, Koltun V. (2018), An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*,.
- Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., ... and Bengio Y. (2014), Generative adversarial nets. *Advances in neural information processing systems*,, **27**.
- Heaton J. Goodfellow I, Bengio Y. and Courville A. (2016), Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618. *Genetic programming and evolvable machines*,, **19(1)**, 305-307.
- Vuletić M., Prenzel F., and Cucuringu M. (2024), Fin-gan: Forecasting and classifying financial time series via generative adversarial networks. *Quantitative Finance*, **24(2)**, 175-199.
- Wiese M., Knobloch R., Korn R., and Kretschmer P. (2020), Quant GANs: deep generation of financial time series. *Quantitative Finance*, **20(9)**, 1419-1440.