Quant GANs: Deep Generation of Financial Time Series

Hyelin Choi

Department of Mathematics

Sungkyunkwan University

Nov 28, 2024

Table of Contents

Table of Contents

I. Introduction

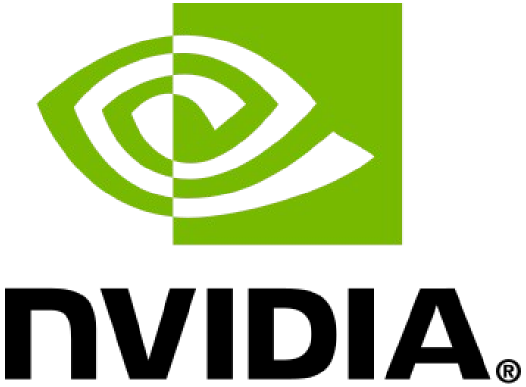
II. Generative Adversarial Networks (GANs)

III. Temporal Convolutional Networks (TCNs)

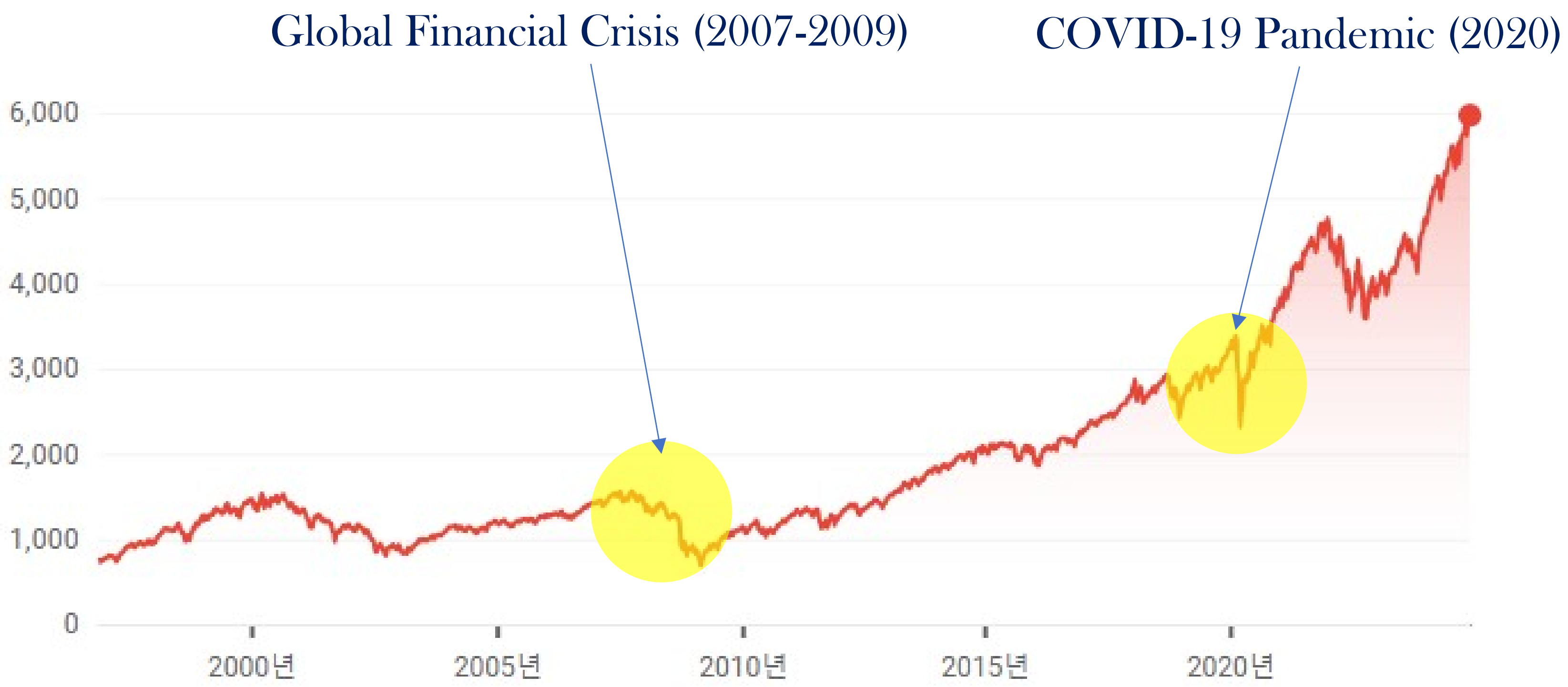
IV. Log-return Neural Process

V. Numerical Results

What is S&P 500?



S&P 500 Stock Price Path



S&P 500 index data, Google Finance

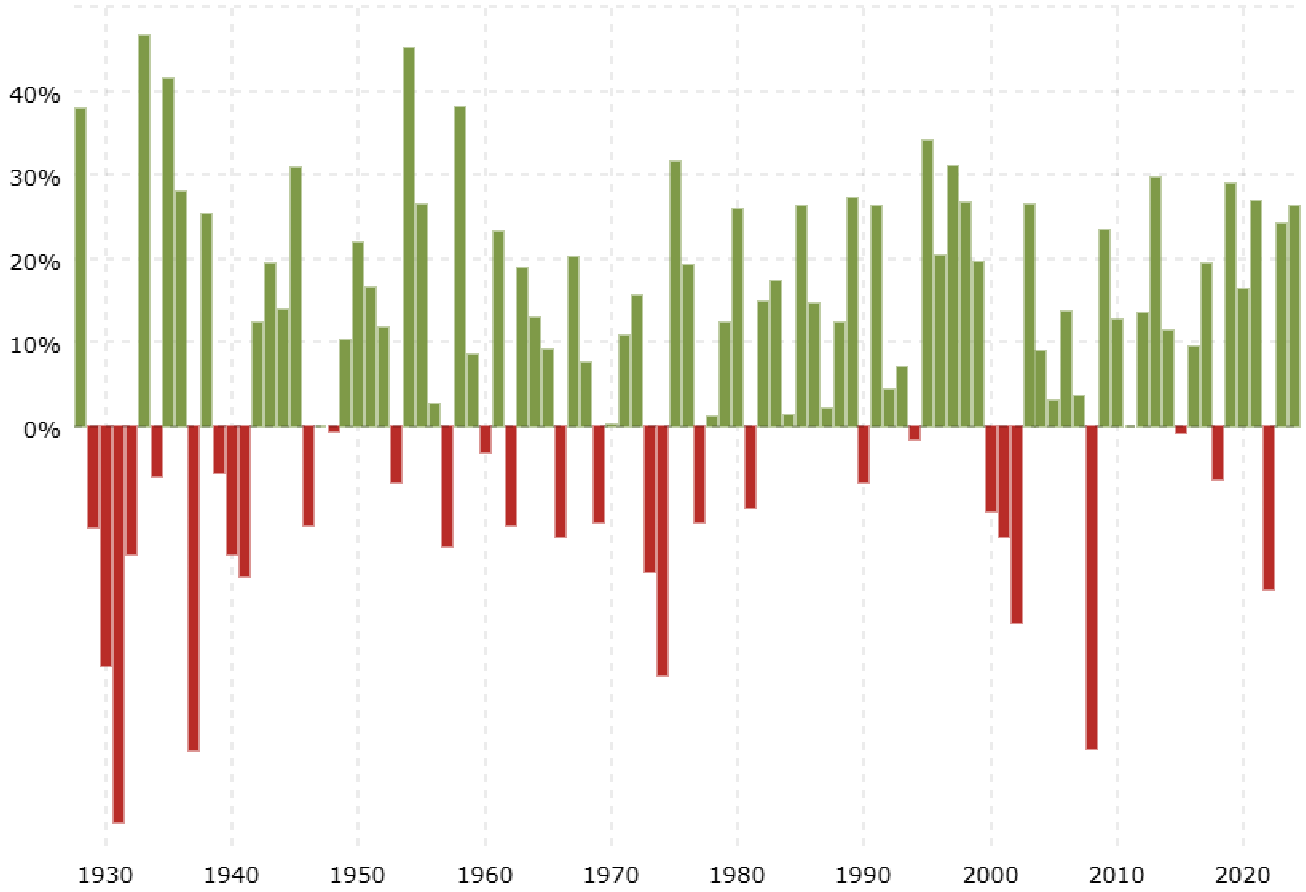
Return

Let be the stock price at time . There are two types of returns.

- Relative return

- Log-return

Introduction

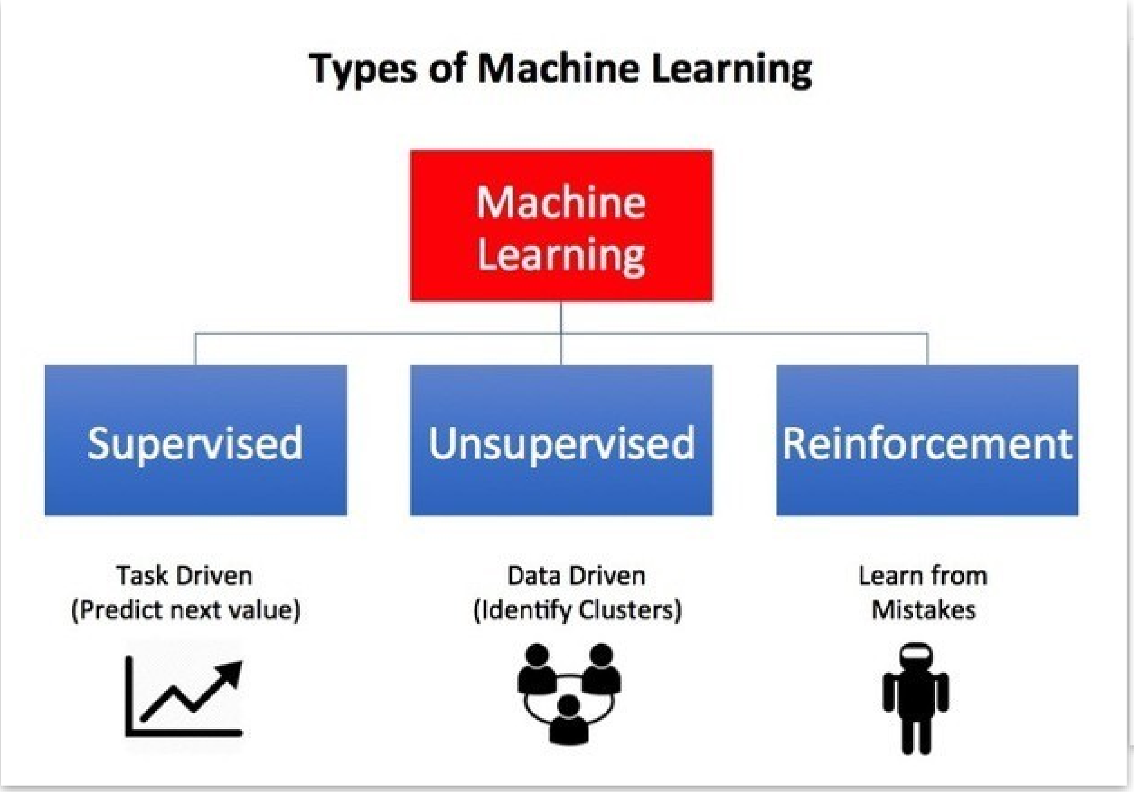


S&P 500 Historical Annual Returns

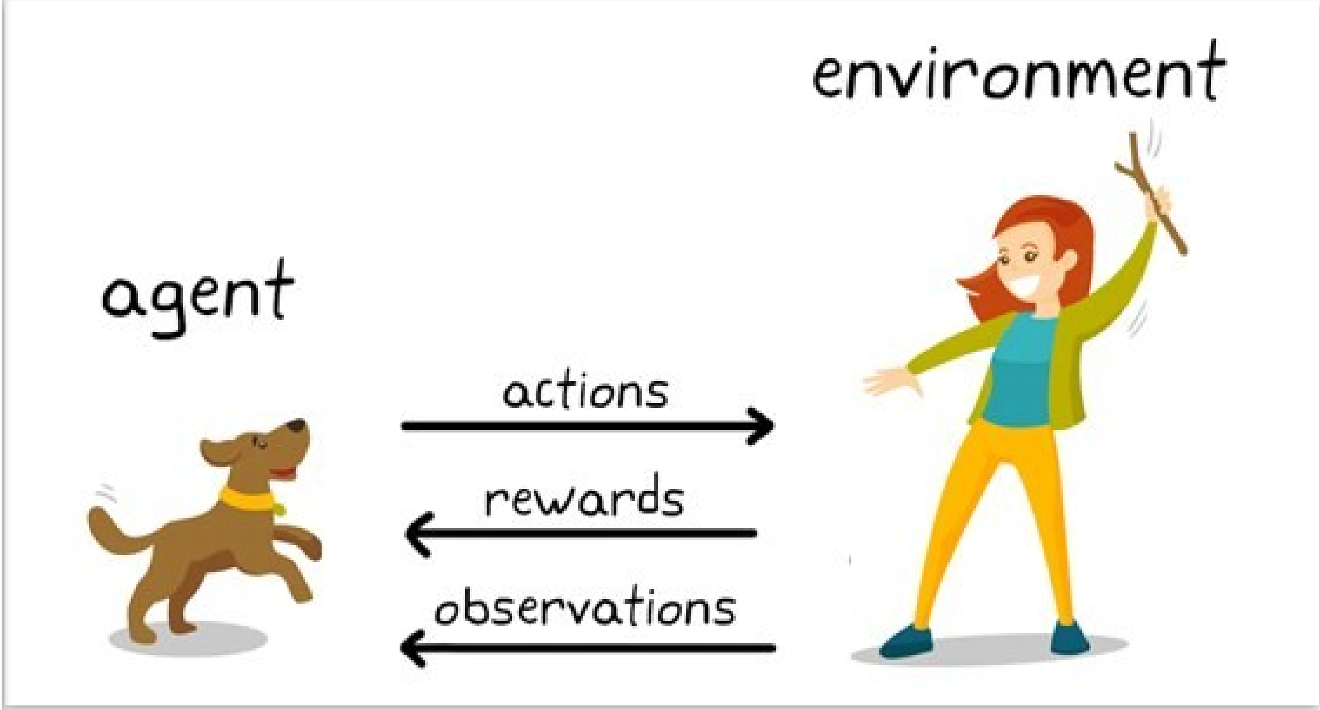
https://www.macrotrends.net/2526/sp-500-historical-annual-returns

Machine Learning in Finance

1. We can create **multiple** plausible future scenarios of asset prices.

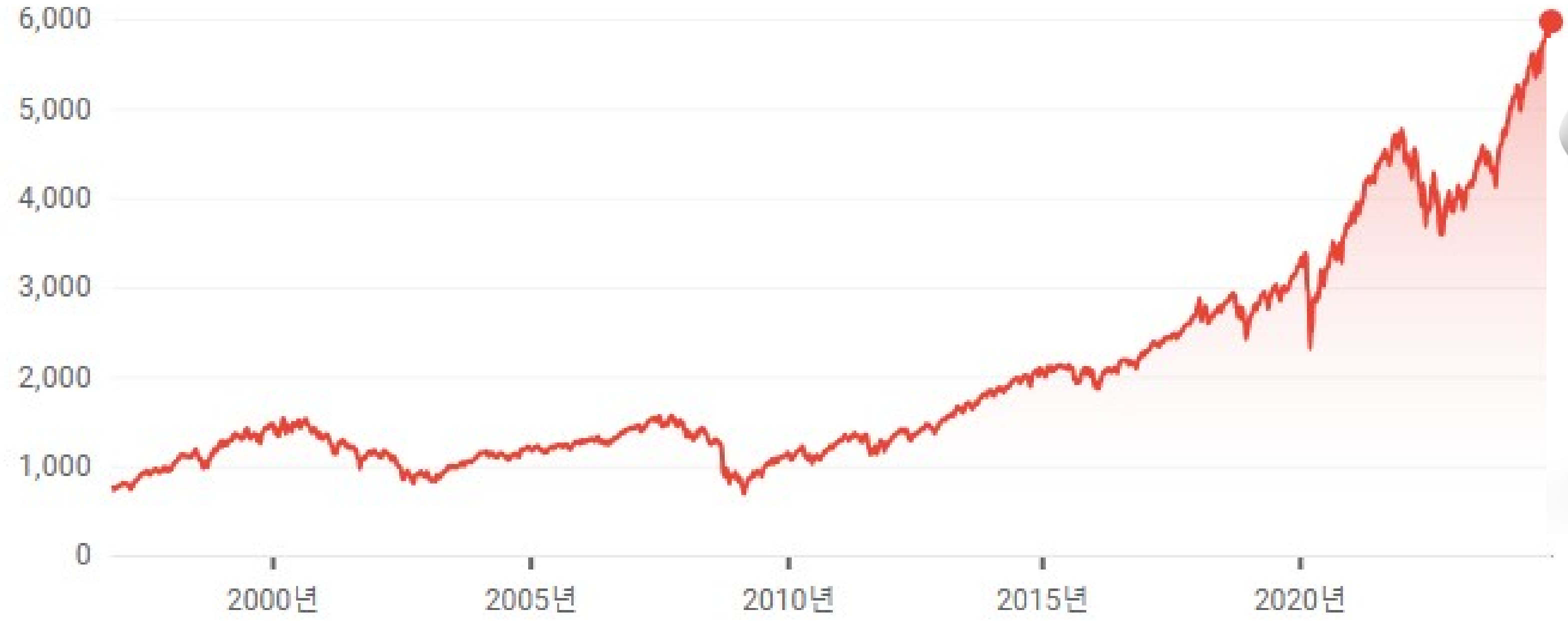


Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement
Task Driven (Predict next value)
Data Driven (Identify Clusters)
Learn from Mistakes



Machine Learning in Finance

2. We can predict **future prices** or **trends** of assets based on past behavior.



QuantGANs

We can generate realistic log-return paths of S&P 500 by using QuantGANs.

QuantGANs is useful as it can be used to extend and enrich unlimited real-world datasets, which in turn can be used to fine-tune or robustify financial trading strategies.

Long-range dependency

What key information should QuantGANs capture from log-return data?

Long-range dependency

1. Leverage effects
2. Volatility clustering
3. Serial autocorrelation

Leverage Effects

1. Leverage effects

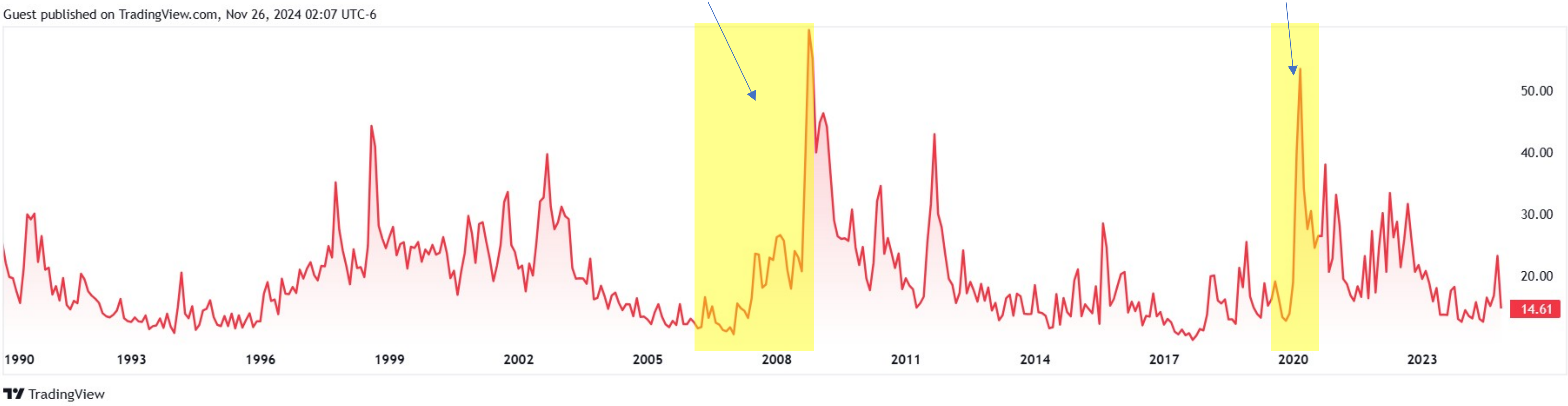
When stock prices experience a significant drop, it is often followed by an increase in volatility.

Volatility Clustering

2. Volatility clustering

Global Finance Crisis between mid 2007 and early 2009

COVID-19 Pandemic (2020)

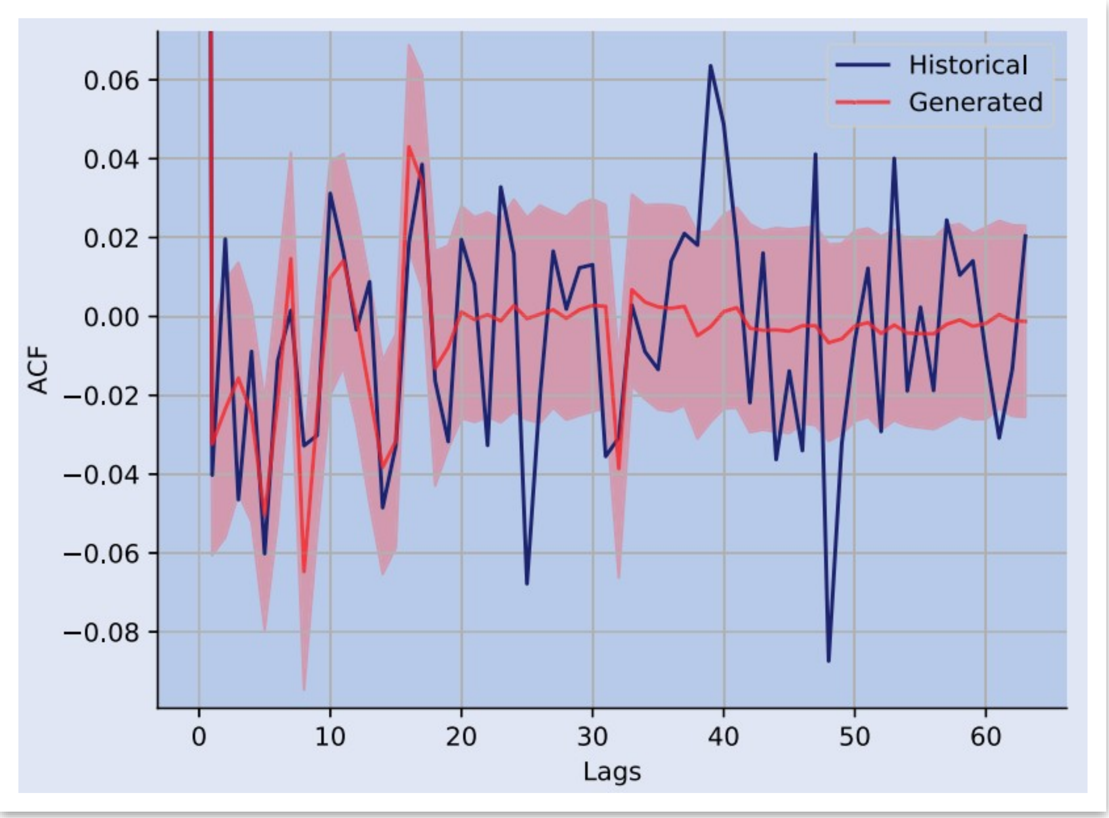


Volatility of S&P 500

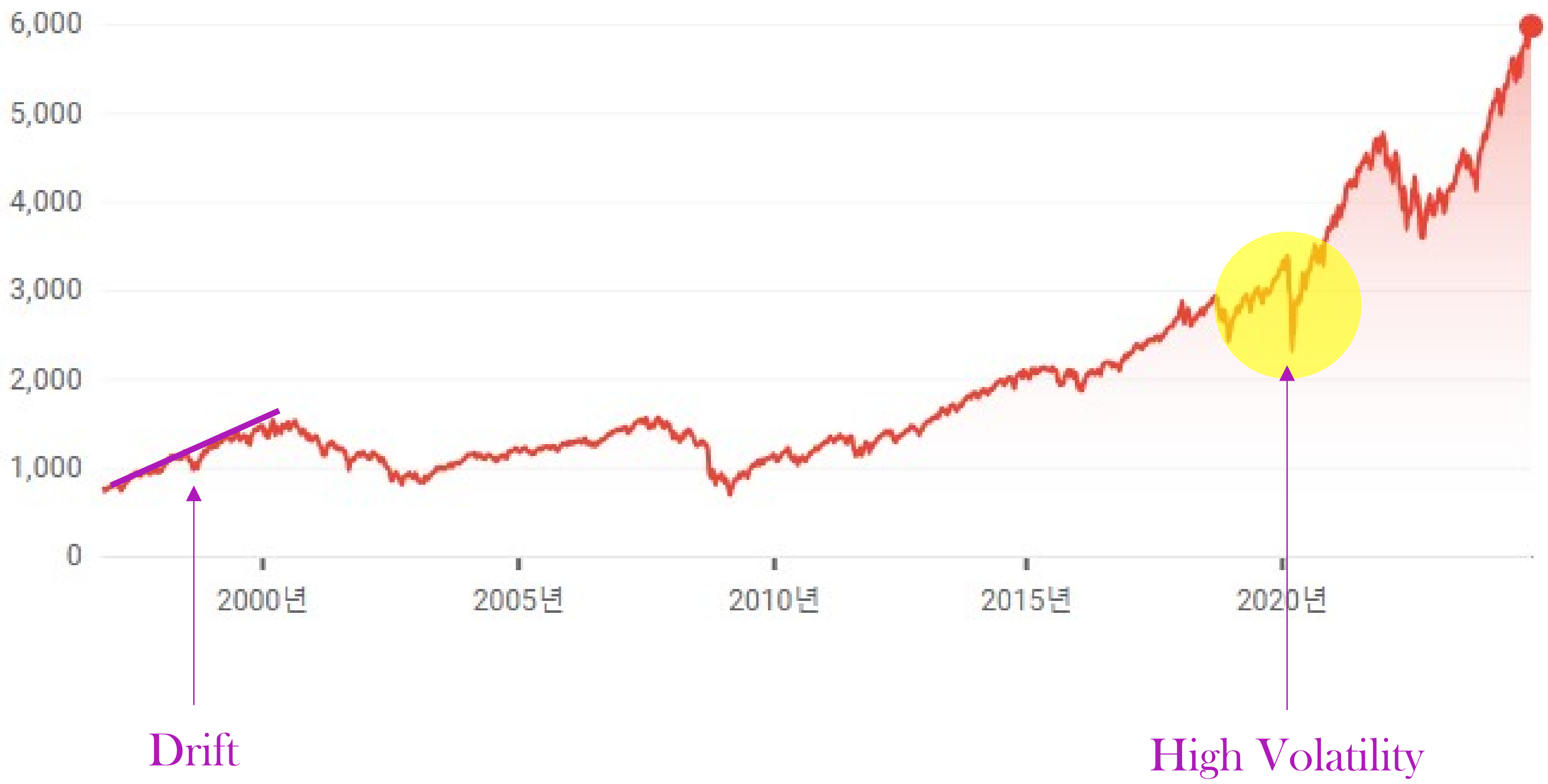
Serial Autocorrelation

3. Serial autocorrelation

- In finance, we use autocorrelation to measure how much influence past prices for a security have on its future price.



Volatility, Drift, and Innovation

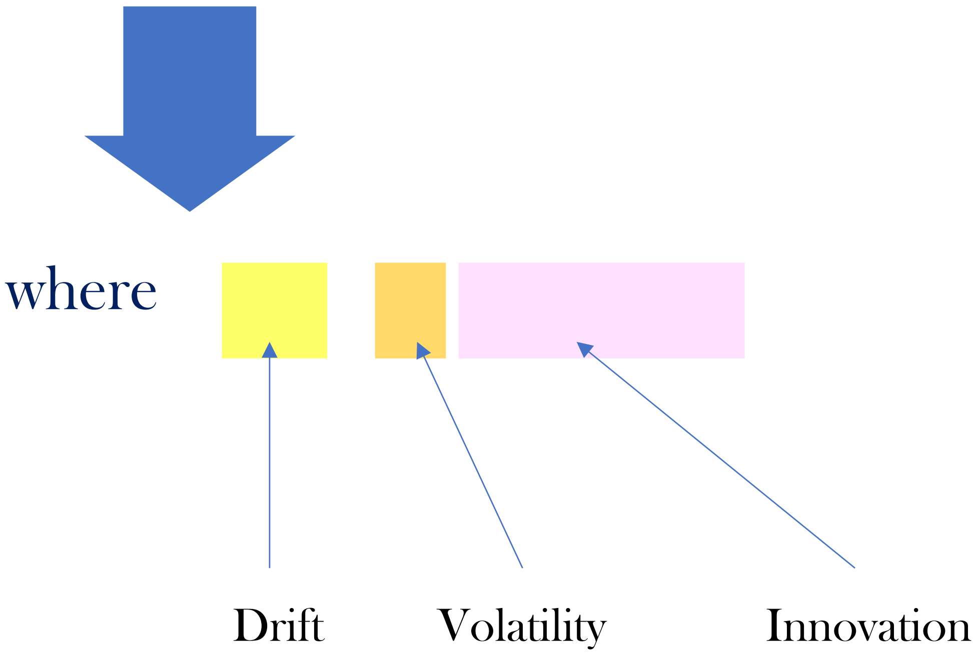


S&P 500 index data, Google Finance

Geometric Brownian Motion(GBM)

Geometric Brownian Motion (GBM) is a mathematical model used to describe the random behavior of asset prices over time.

where : Brownian motion, : drift and : volatility



where

Drift

Volatility

Innovation

Table of Contents

I. Introduction

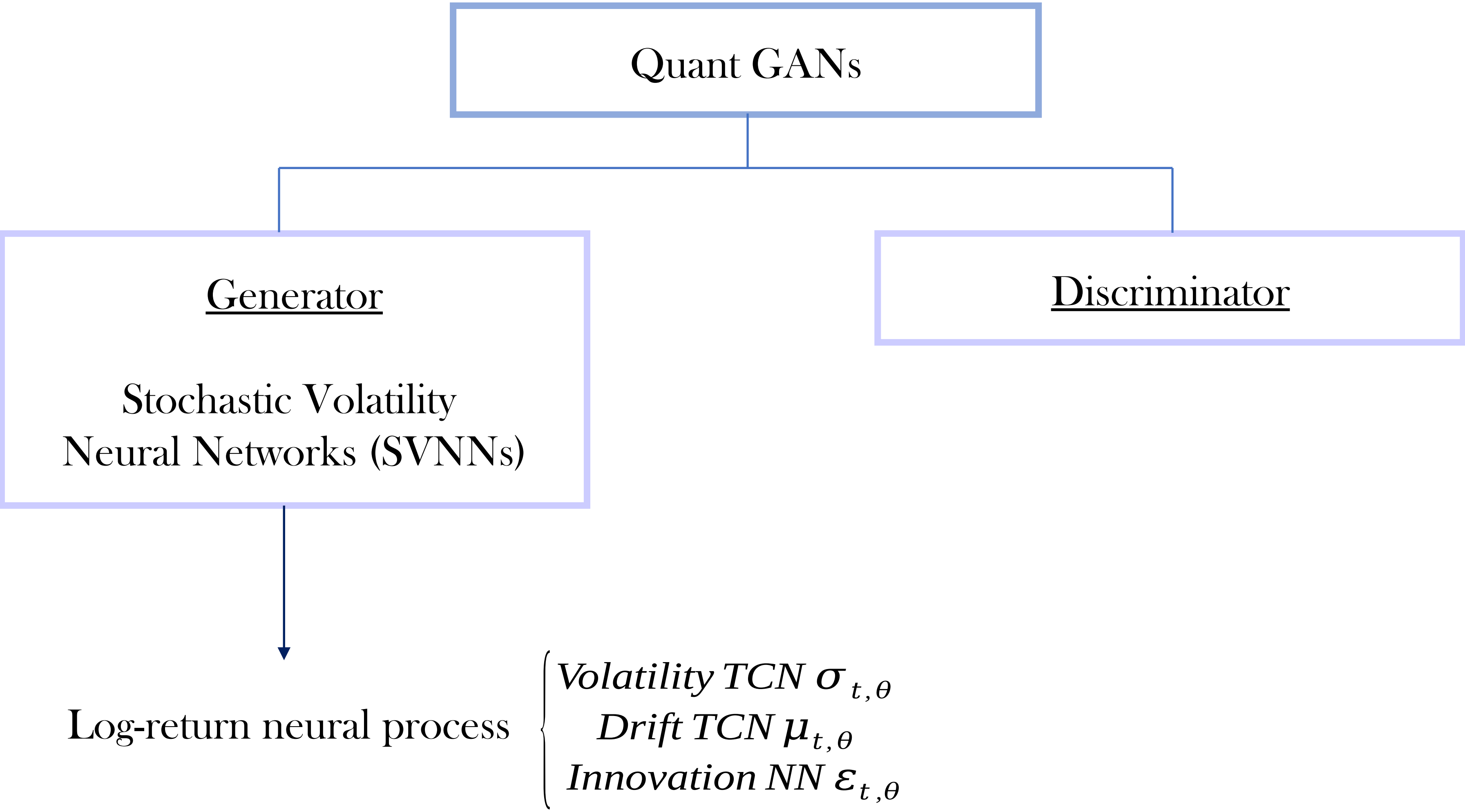
II. Generative Adversarial Networks (GANs)

III. Temporal Convolutional Networks (TCNs)

IV. Log-return Neural Process

V. Numerical Results

Construction of QuantGANs



Quant GANs

<u>Generator</u>

Stochastic Volatility
Neural Networks (SVNNs)

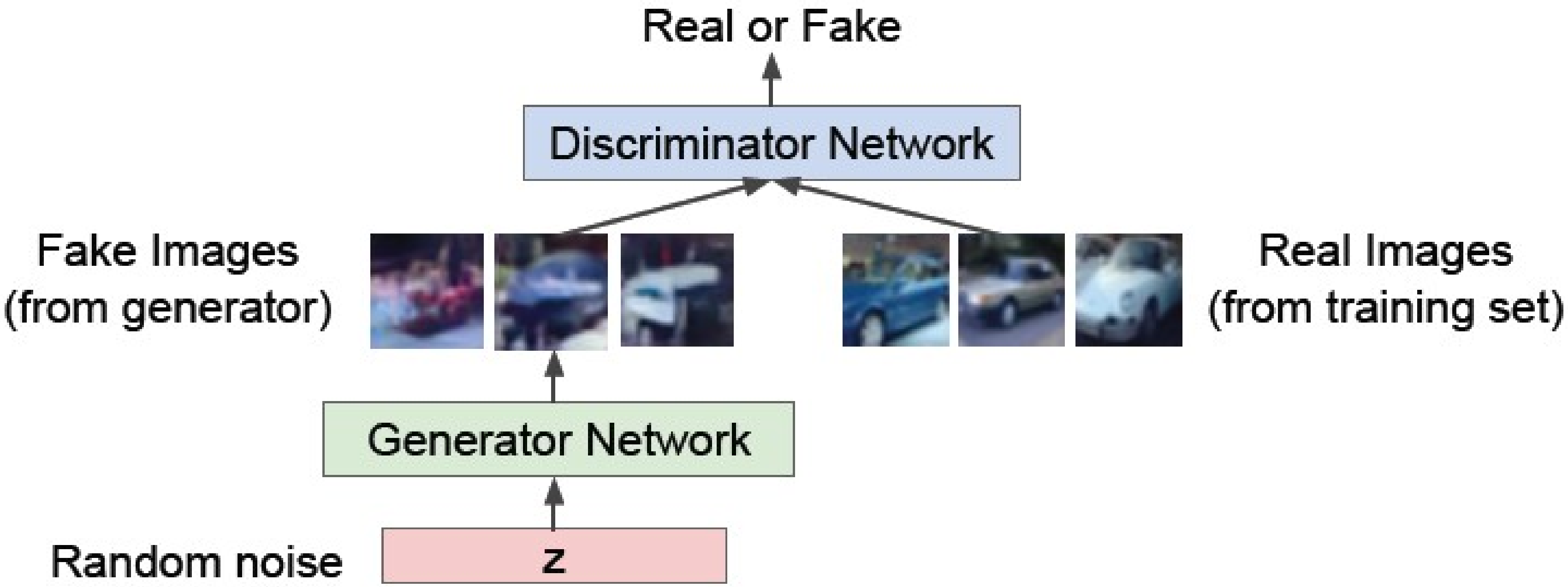
<u>Discriminator</u>

Log-return neural process

$$\begin{cases} \textit{Volatility TCN } \sigma_{t,\theta} \\ \textit{Drift TCN } \mu_{t,\theta} \\ \textit{Innovation NN } \varepsilon_{t,\theta} \end{cases}$$

Generative Adversarial Networks

대립적인



Yunhuijang, 2022.7.20, https://process-mining.tistory.com/169

Generative Adversarial Networks

i.i.d. Gaussian noise process

- The random variable **Z** represents the **noise** prior and **X** the **targeted** (or data) random variable.

- The goal of GANs is to train a network such that the induced random variable for some parameter and the targeted random variable have the same distribution, i.e. .

d

Generative Adversarial Networks

The **generator** aims at generating samples such that the discriminator cannot distinguish whether the realizations were sampled from the target or the generator distribution. In other words, the **discriminator** acts as a classifier that assigns to each sample a probability of being a realization of the target distribution.

Generative Adversarial Networks

Loss function of GANs

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

: parameter of discriminator
: parameter of generator
: function of discriminator
: targeted r.v.
: generated r.v.

Step 1

Let the 1 represent real data and 0 represent fake data.
The discriminator's parameter are chosen to maximize the function .

Step 2

The generator's parameters are trained to minimized the probability of generated samples being identified as such and not from the data distribution.

Generative Adversarial Networks

We get the min-max problem

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

which refer to as the GAN objective.

Table of Contents

I. Introduction

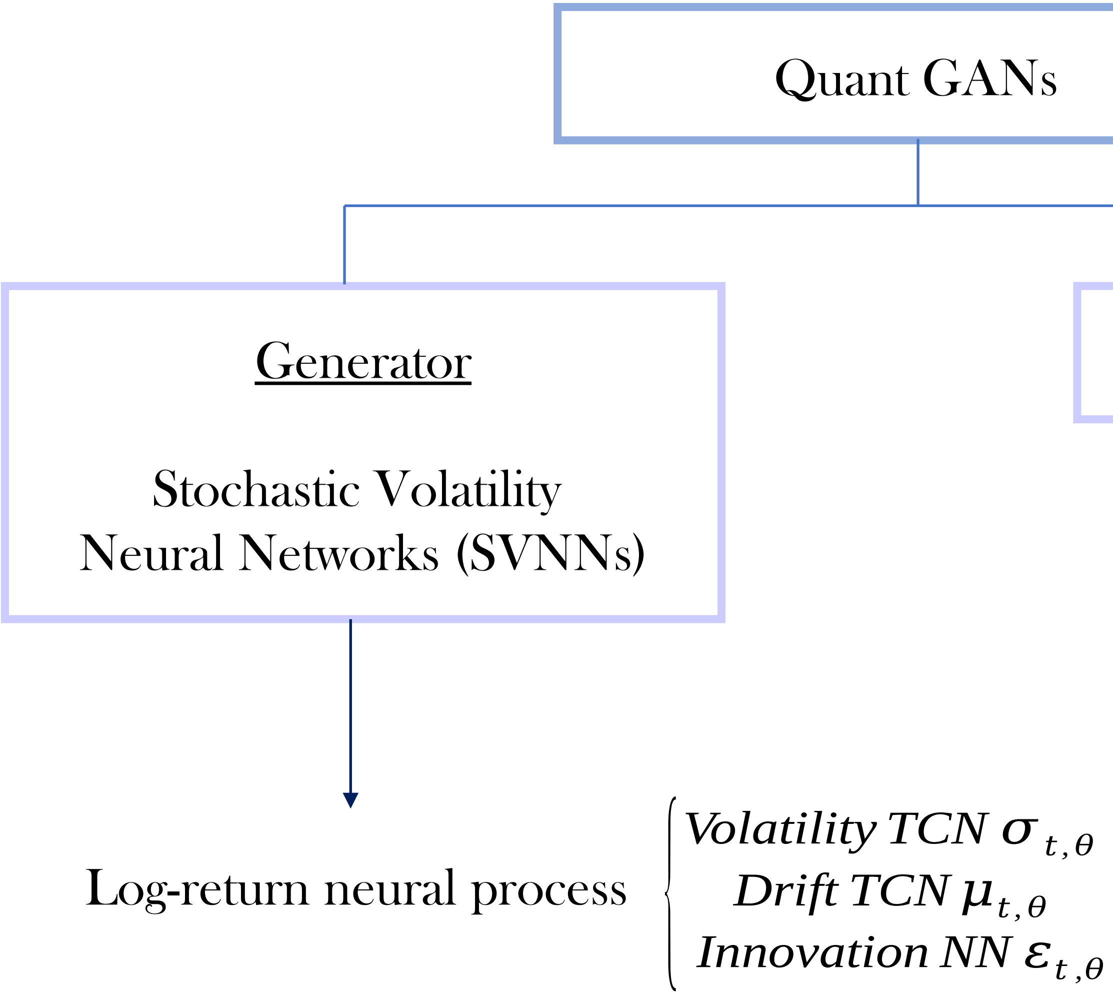
II. Generative Adversarial Networks (GANs)

III. Temporal Convolutional Networks (TCNs)

IV. Log-return Neural Process

V. Numerical Results

Construction of QuantGANs



Quant GANs

Generator

Stochastic Volatility Neural Networks (SVNNs)

Discriminator

Log-return neural process $\begin{cases} \textit{Volatility TCN } \sigma_{t,\theta} \\ \textit{Drift TCN } \mu_{t,\theta} \\ \textit{Innovation NN } \varepsilon_{t,\theta} \end{cases}$

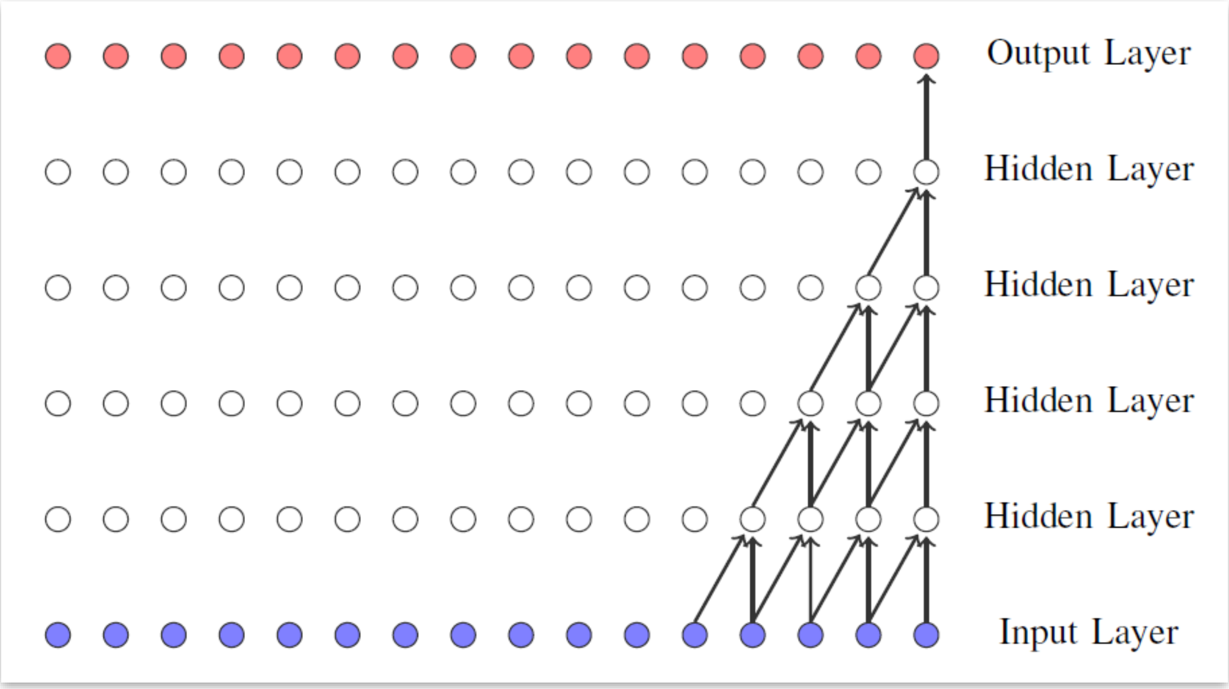
Volatility and drift are modeled with TCN to capture long-range dependencies.

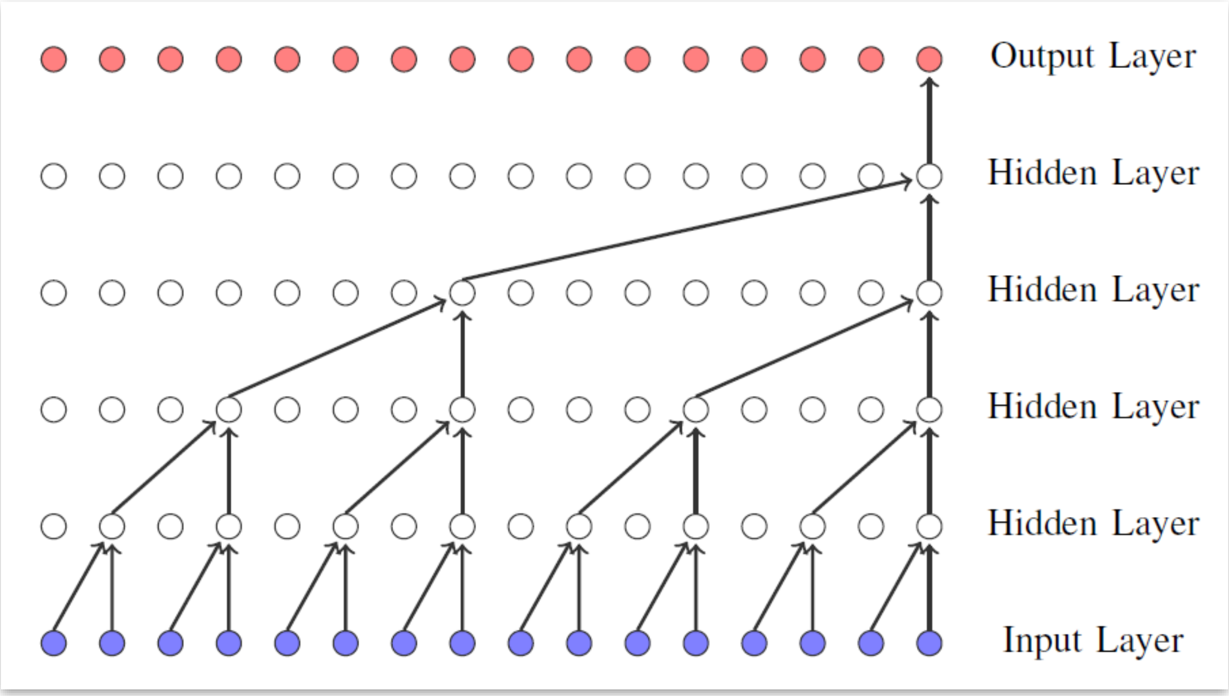
Why We Use TCNs for Long-range Dependency

1) TCNs are able to capture long-range dependencies in sequences.

2) TCNs have an advantage of avoiding exponentially vanishing and exploding gradients.

Why We Use TCNs for Long-range Dependency





Construction of TCNs

TCNs are neural network models primarily designed to efficiently handle sequential data, such as time series.

- Constructions

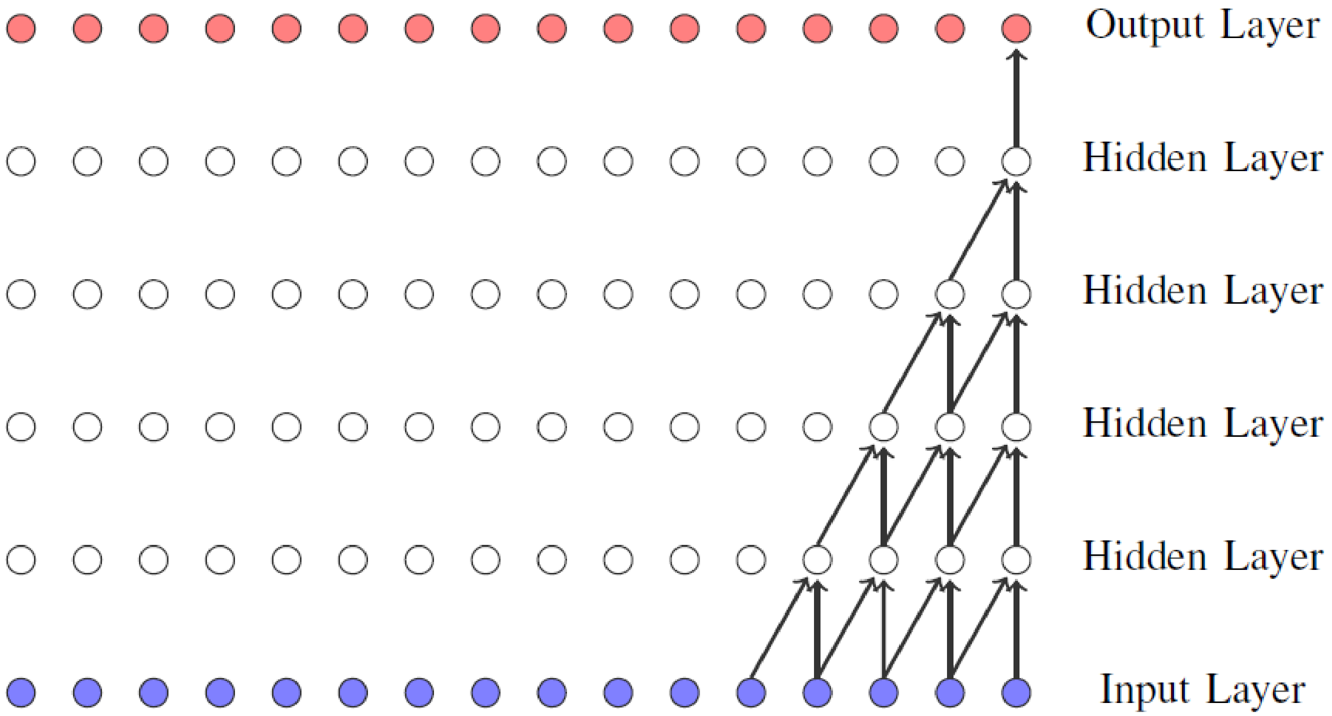
Dilated causal convolutions = Causal convolutions + Dilated convolutions

인과 확장

Causal Convolution

Causal convolutions are convolutions, where output only depends on past sequence elements.

인과

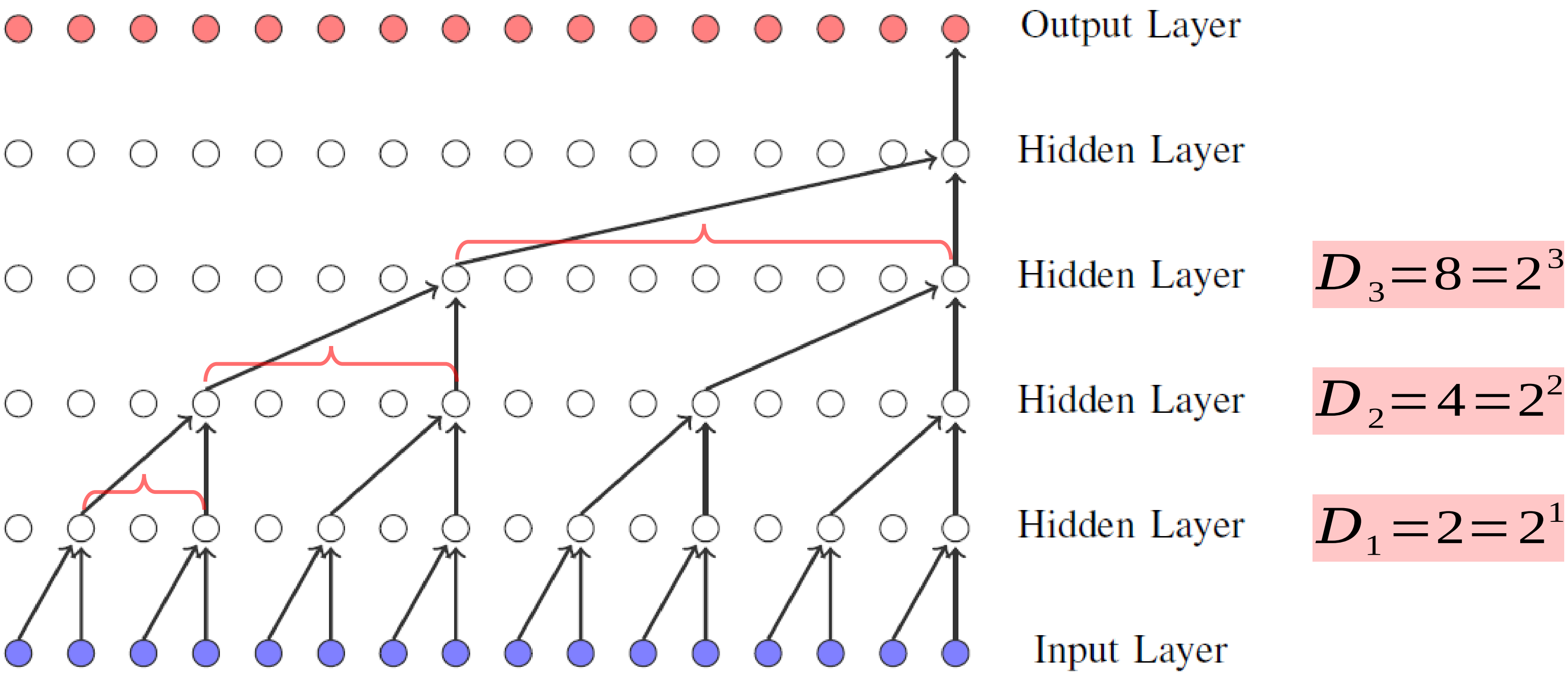


Dilated Convolution

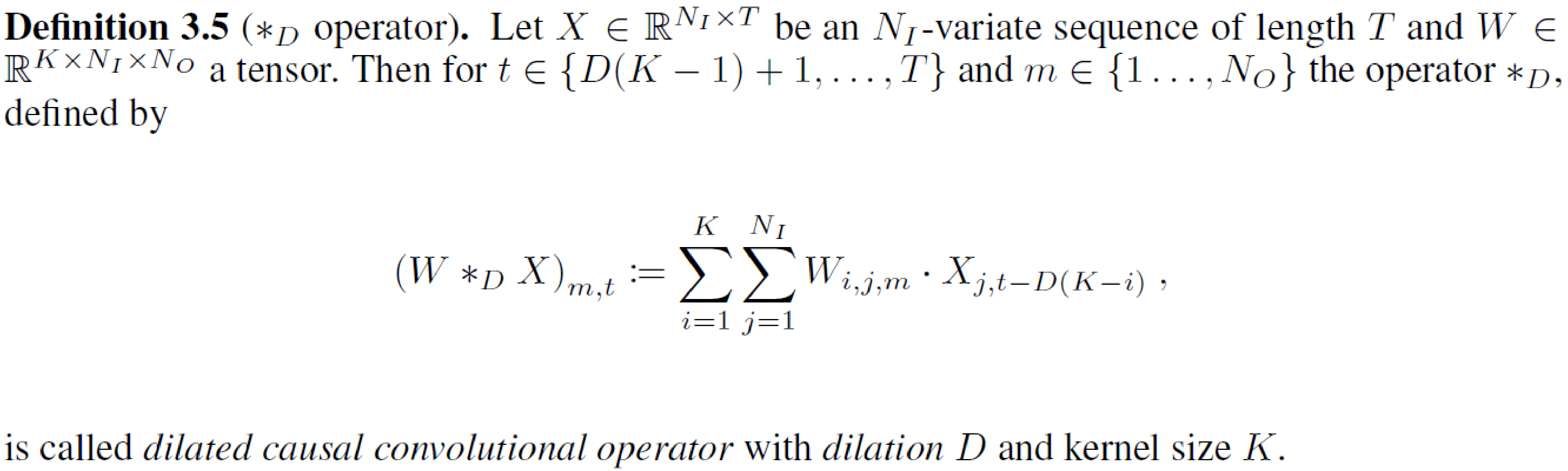
Dilated convolutions are convolutions ‘with holes’.

확장

- Dilation factor D
- Kernel size K=2



Operator

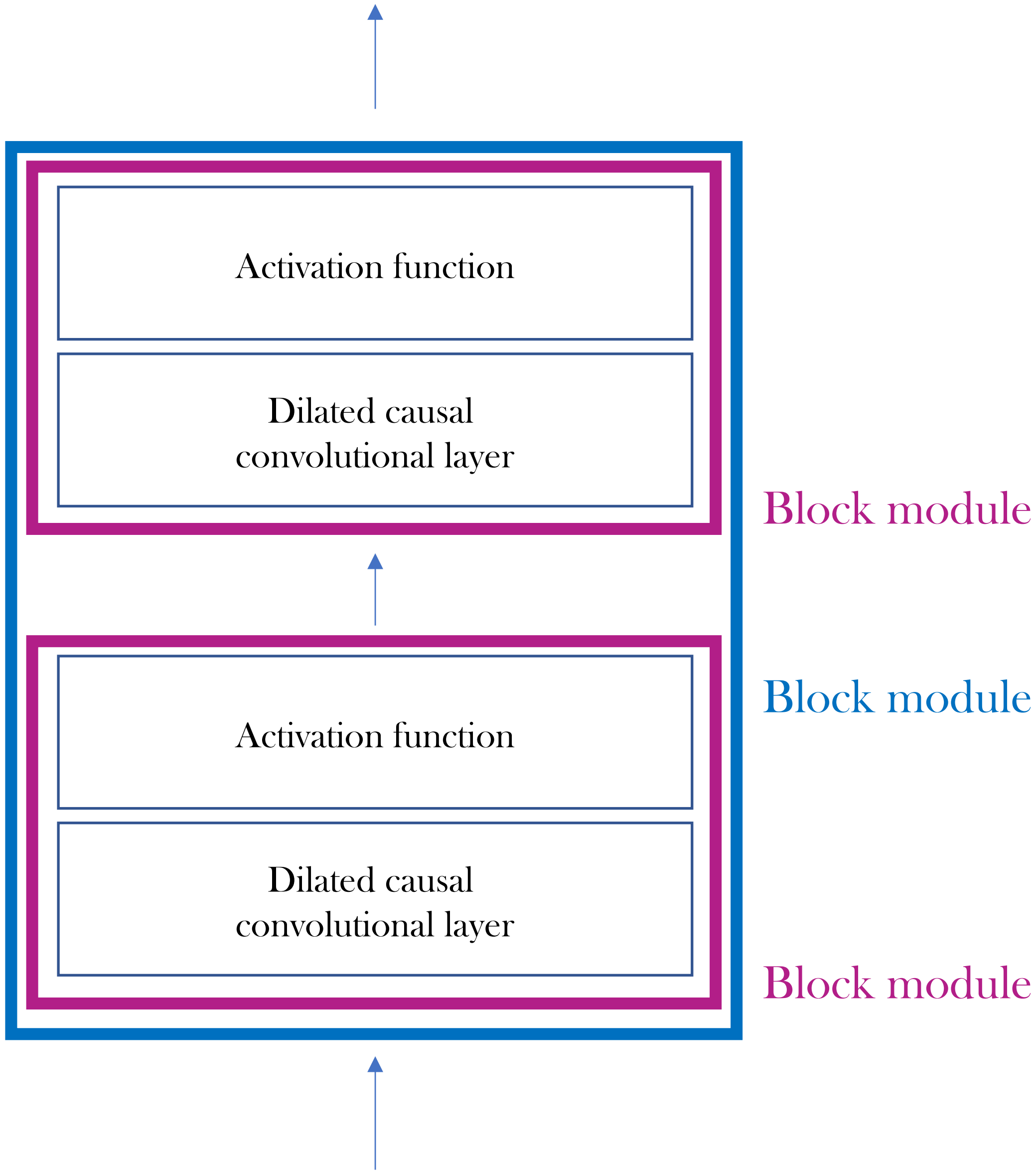


Definition 3.5 ($*_D$ operator)**.** Let $X \in \mathbb{R}^{N_I \times T}$ be an N_I-variate sequence of length T and $W \in \mathbb{R}^{K \times N_I \times N_O}$ a tensor. Then for $t \in \{D(K-1)+1,\dots,T\}$ and $m \in \{1\dots,N_O\}$ the operator $*_D$, defined by

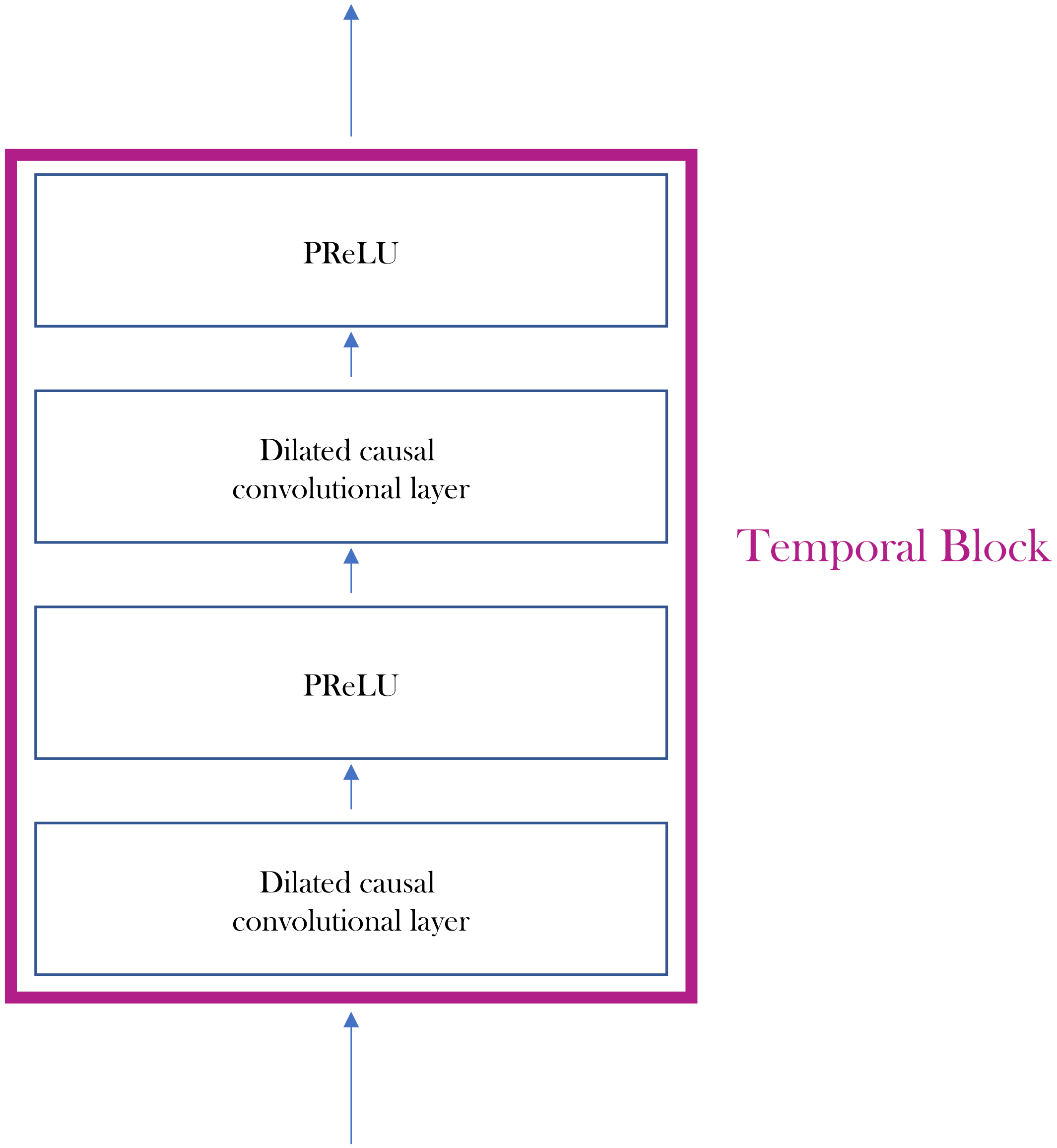
$$(W *_D X)_{m,t} := \sum_{i=1}^{K} \sum_{j=1}^{N_I} W_{i,j,m} \cdot X_{j,t-D(K-i)} ,$$

is called *dilated causal convolutional operator* with *dilation* D and kernel size K.

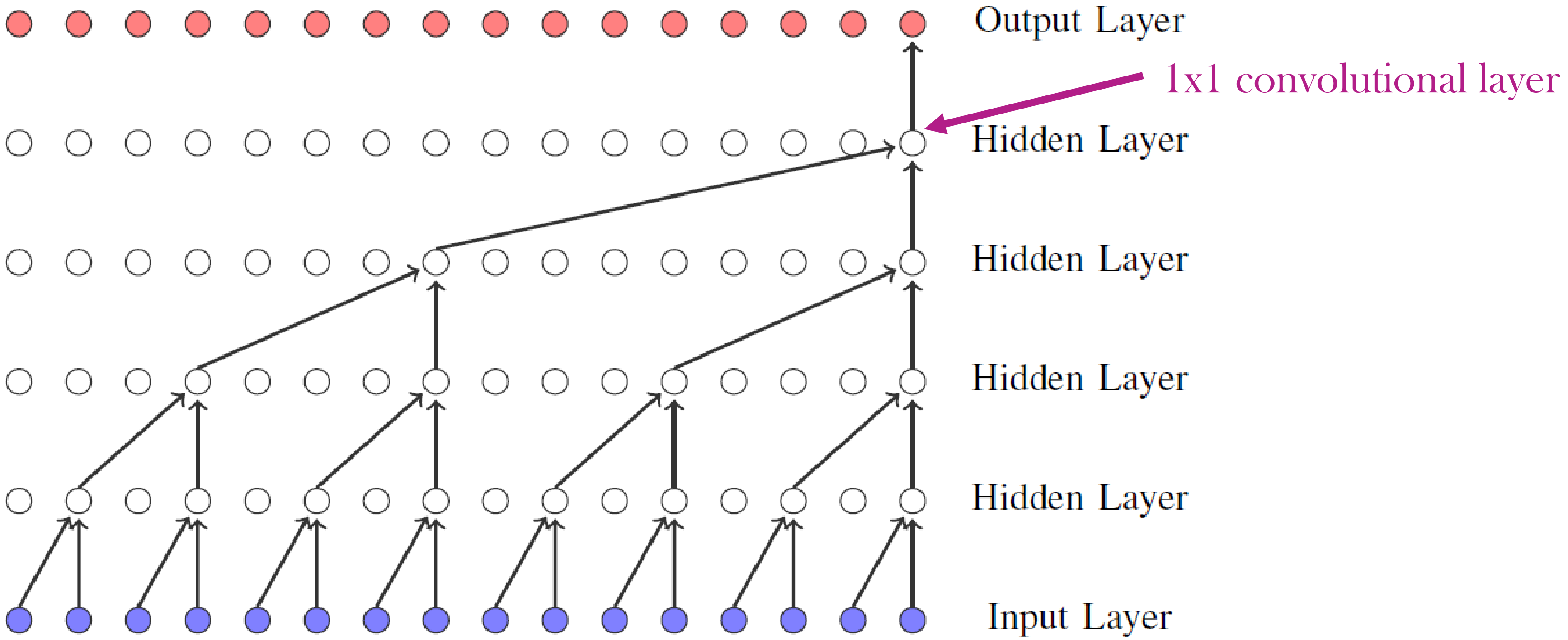
Block Module



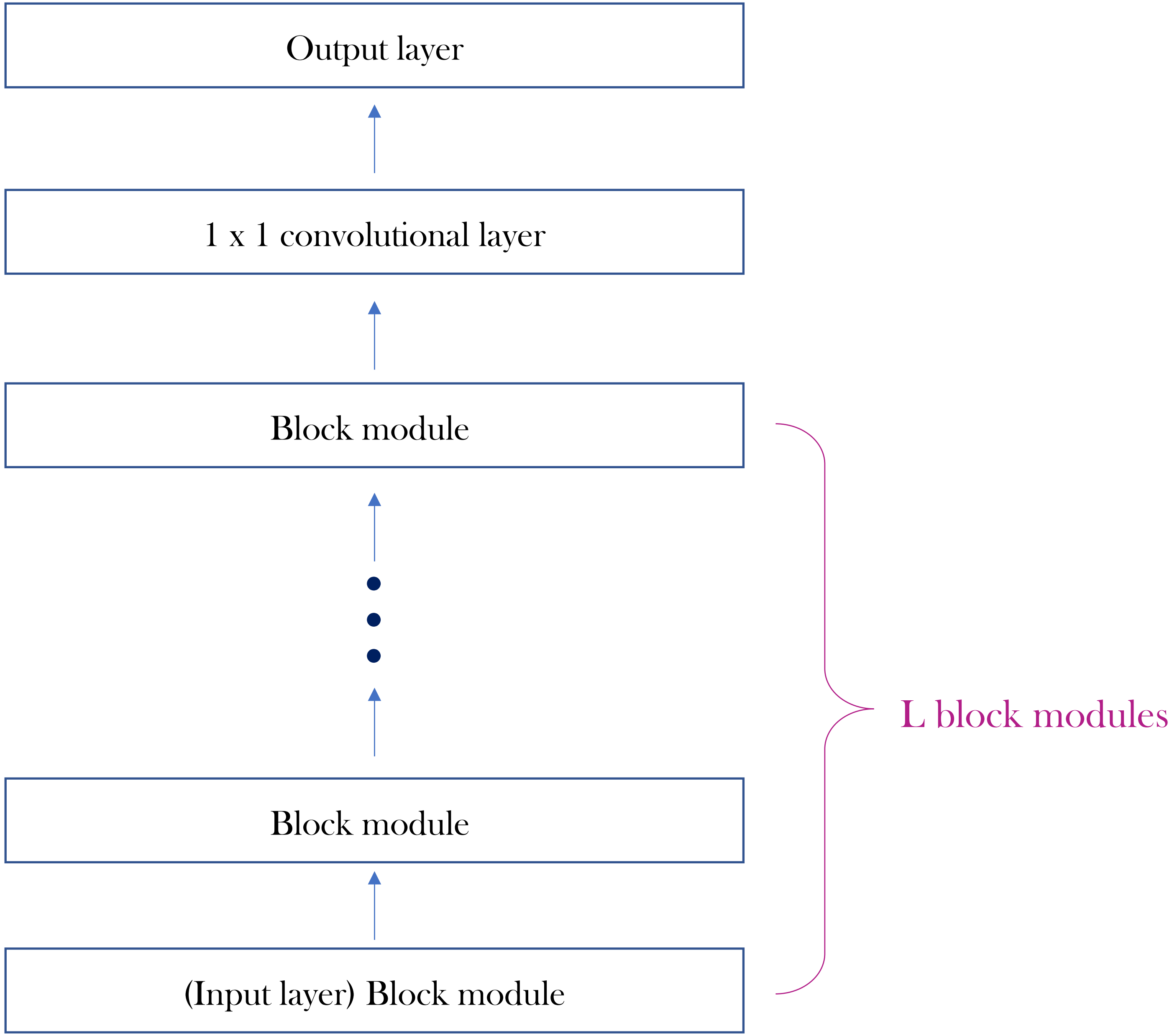
ex)


PReLU
Dilated causal convolutional layer
Temporal Block
PReLU
Dilated causal convolutional layer

1x1 Convolutional Layer



Temporal Convolutional Network



Output layer

1 x 1 convolutional layer

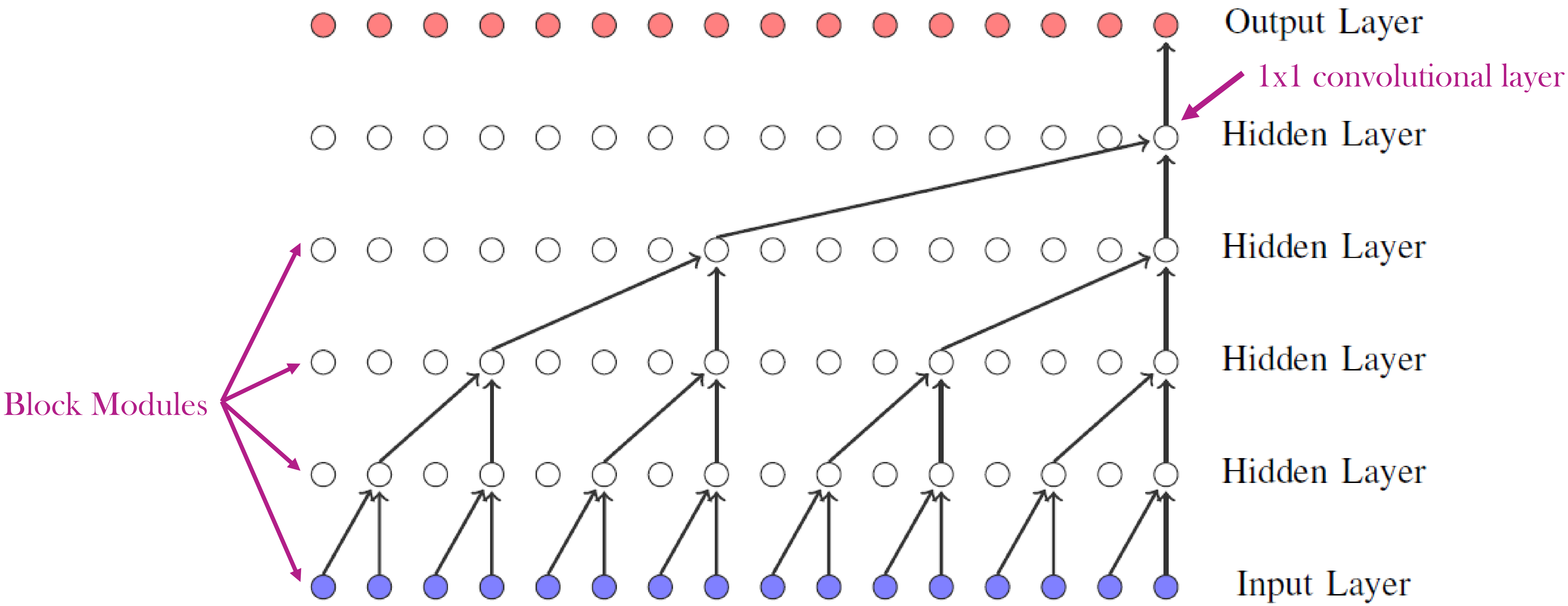
Block module

Block module

(Input layer) Block module

L block modules

Temporal Convolutional Networks



Skip Connections

Skip Connections

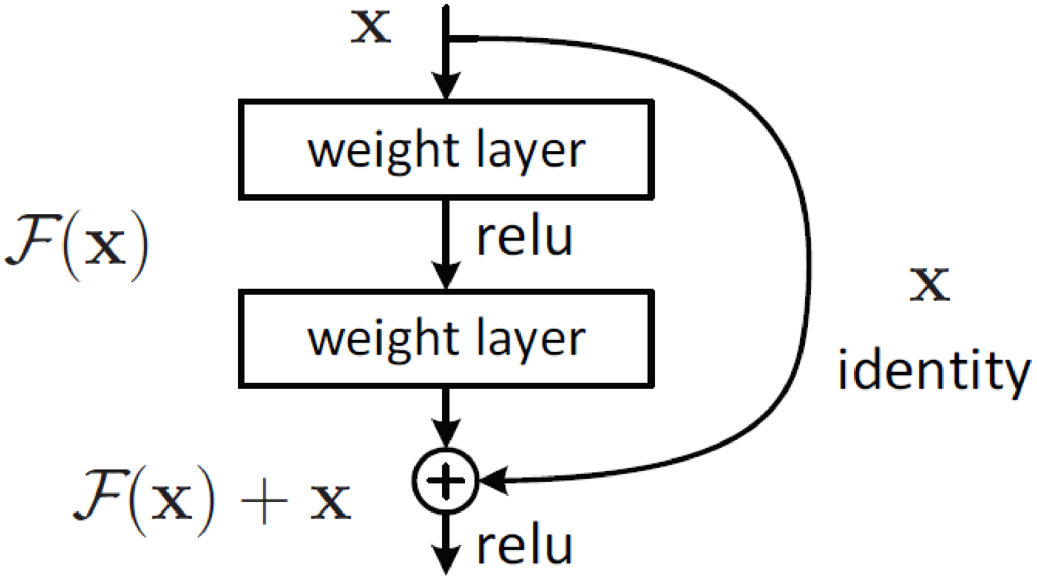
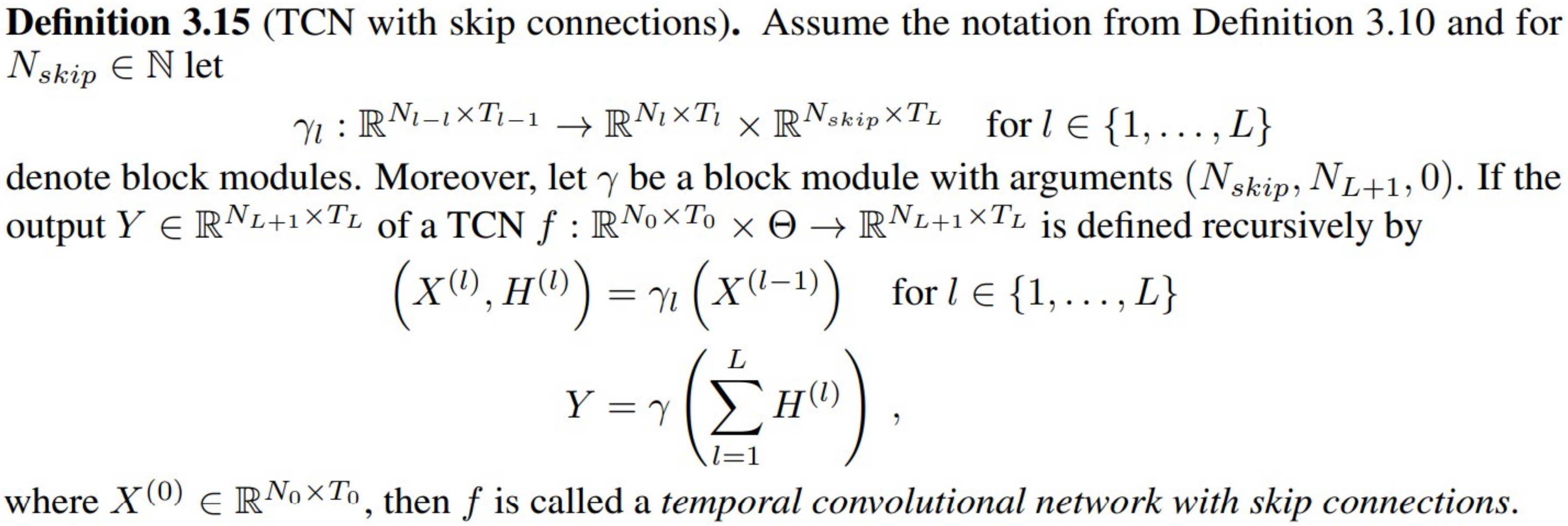

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
x
weight layer
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

TCN with Skip Connections



Definition 3.15 (TCN with skip connections)**.** Assume the notation from Definition 3.10 and for $N_{skip} \in \mathbb{N}$ let

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

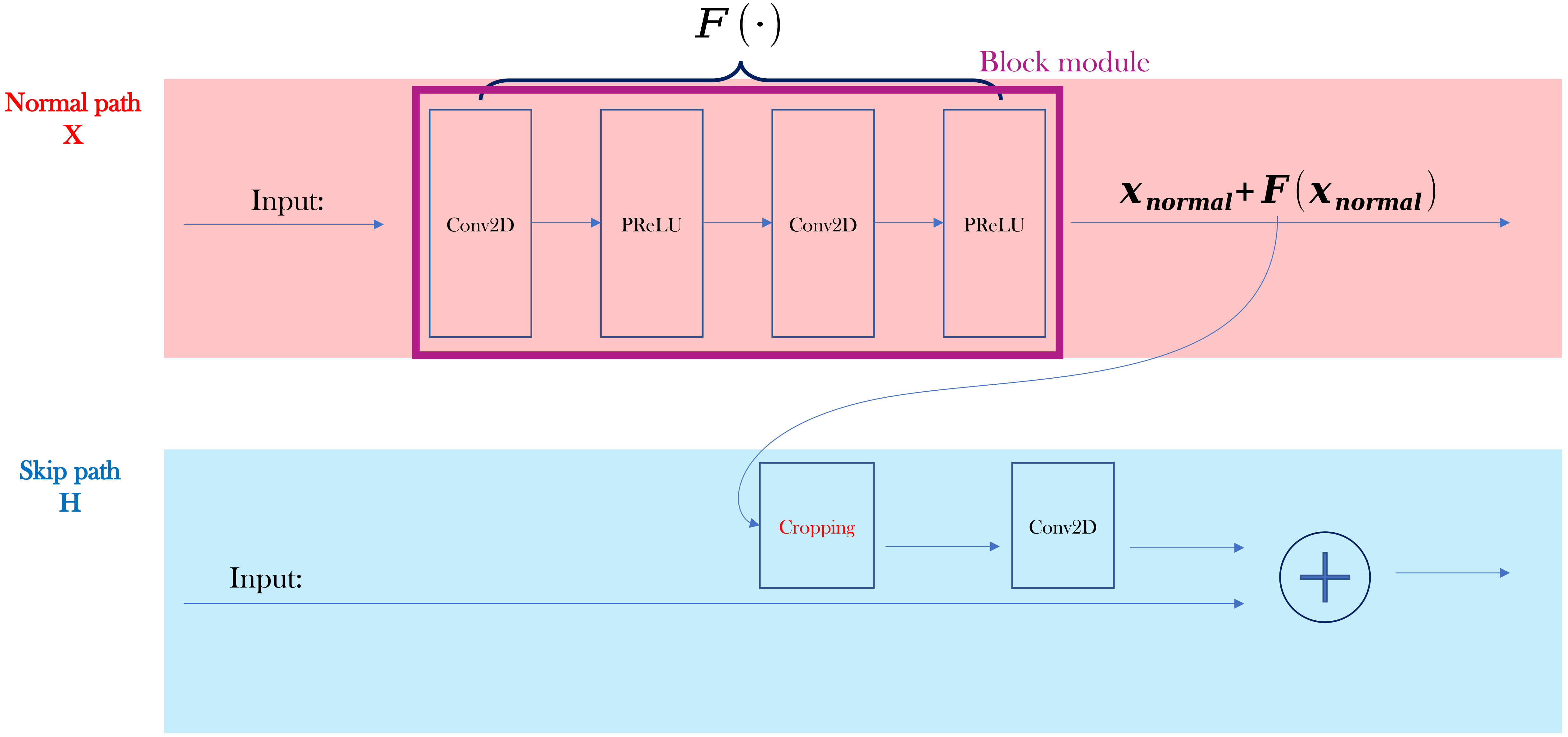
denote block modules. Moreover, let γ be a block module with arguments $(N_{skip}, N_{L+1}, 0)$. If the output $Y \in \mathbb{R}^{N_{L+1} \times T_L}$ of a TCN $f : \mathbb{R}^{N_0 \times T_0} \times \Theta \to \mathbb{R}^{N_{L+1} \times T_L}$ is defined recursively by

$$\left(X^{(l)}, H^{(l)}\right) = \gamma_l \left(X^{(l-1)}\right) \quad \text{for } l \in \{1, \dots, L\}$$

$$Y = \gamma \left(\sum_{l=1}^{L} H^{(l)} \right) ,$$

where $X^{(0)} \in \mathbb{R}^{N_0 \times T_0}$, then f is called a *temporal convolutional network with skip connections*.

TCN with Skip Connections



Vanilla TCN with Skip Connection

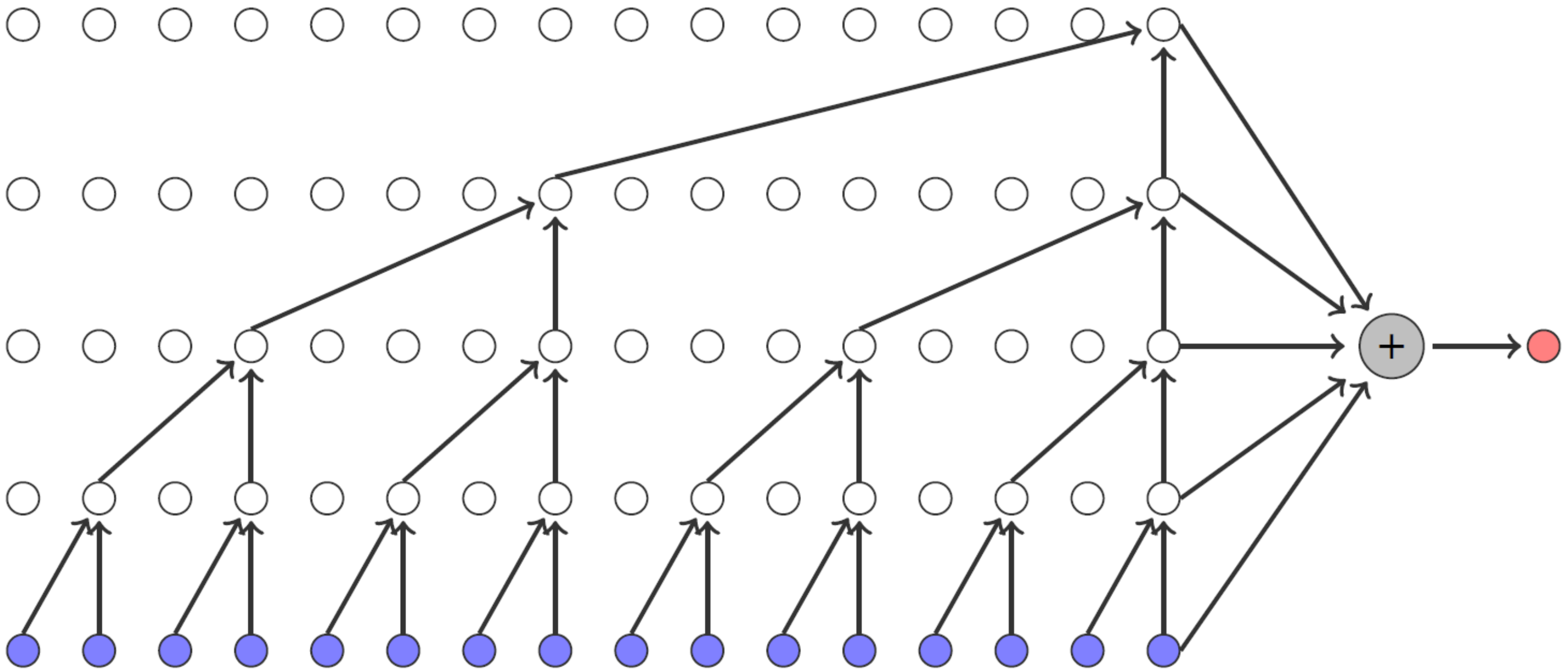


Figure 7: Vanilla TCN with skip connections.

Table of Contents

I. Introduction

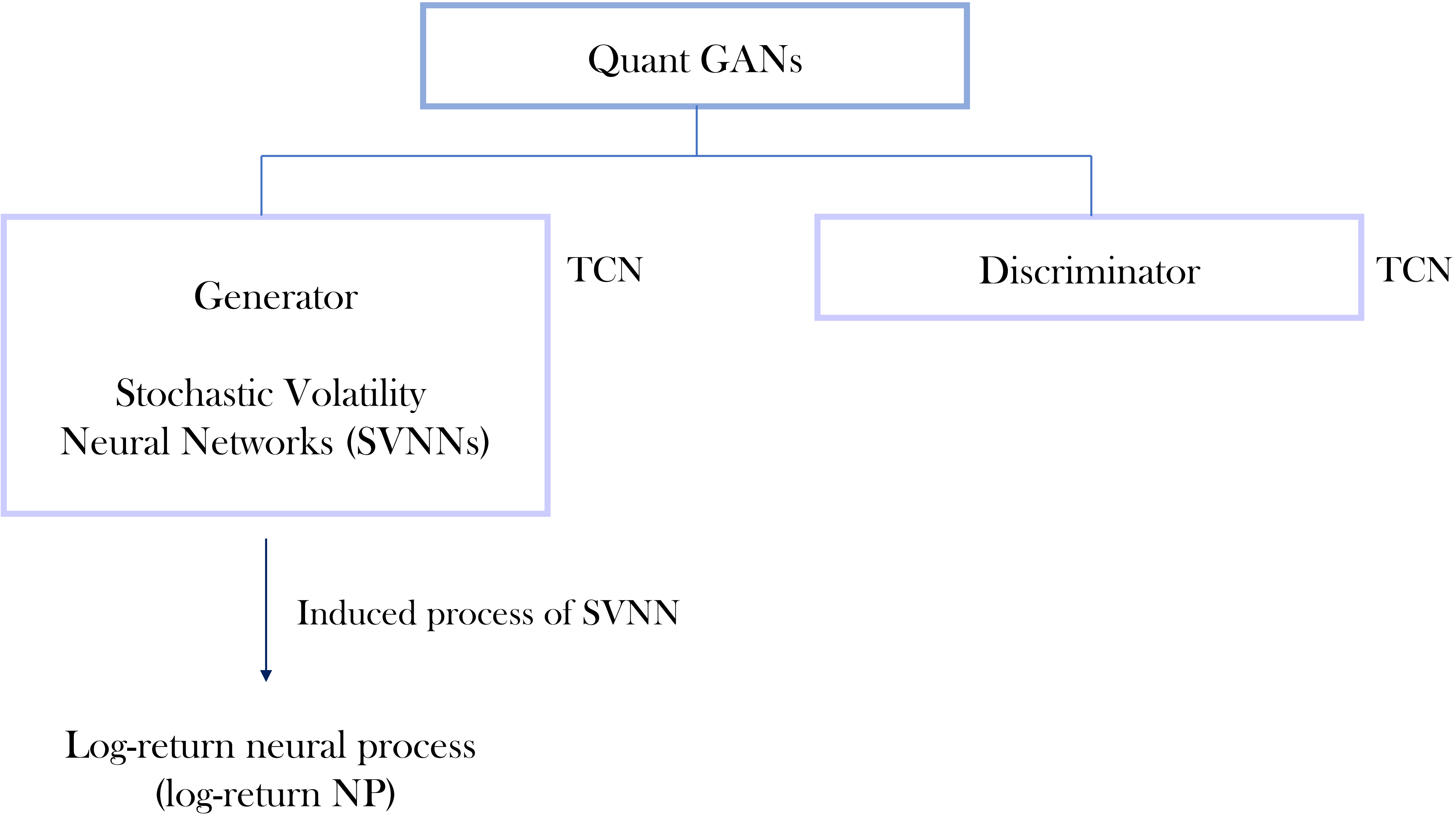
II. Generative Adversarial Networks (GANs)

III. Temporal Convolutional Networks (TCNs)

IV. Log-return Neural Process

V. Numerical Results

Overview



Quant GANs

Generator

Stochastic Volatility Neural Networks (SVNNs)

TCN

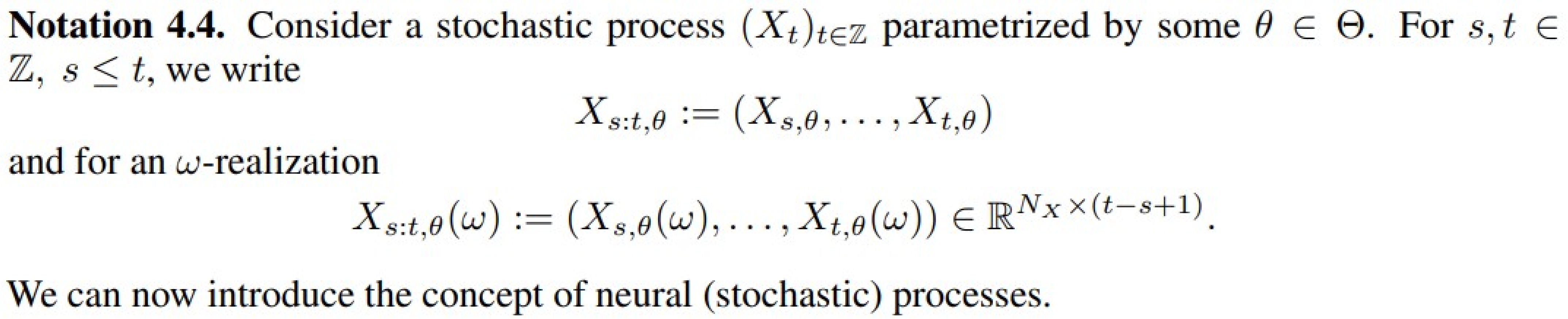
Discriminator

TCN

Induced process of SVNN

Log-return neural process (log-return NP)

Log-return Neural Process



Notation 4.4. Consider a stochastic process $(X_t)_{t\in\mathbb{Z}}$ parametrized by some $\theta \in \Theta$. For $s, t \in \mathbb{Z},\ s \leq t$, we write

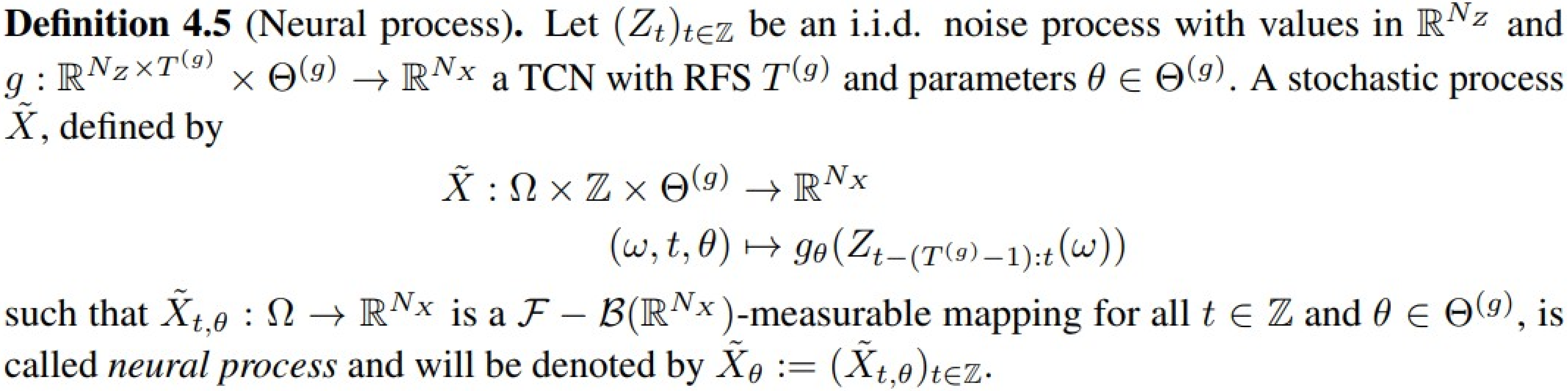
$$X_{s:t,\theta} := (X_{s,\theta}, \dots, X_{t,\theta})$$

and for an ω-realization

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

We can now introduce the concept of neural (stochastic) processes.

Log-return Neural Process



Definition 4.5 (Neural process). Let $(Z_t)_{t\in\mathbb{Z}}$ be an i.i.d. noise process with values in $\mathbb{R}^{N_Z}$ and $g : \mathbb{R}^{N_Z \times T^{(g)}} \times \Theta^{(g)} \to \mathbb{R}^{N_X}$ a TCN with RFS $T^{(g)}$ and parameters $\theta \in \Theta^{(g)}$. A stochastic process $\tilde{X}$, defined by

$$\begin{aligned} \tilde{X} : \Omega \times \mathbb{Z} \times \Theta^{(g)} &\to \mathbb{R}^{N_X} \\ (\omega, t, \theta) &\mapsto g_\theta(Z_{t-(T^{(g)}-1):t}(\omega)) \end{aligned}$$

such that $\tilde{X}_{t,\theta} : \Omega \to \mathbb{R}^{N_X}$ is a $\mathcal{F} - \mathcal{B}(\mathbb{R}^{N_X})$-measurable mapping for all $t \in \mathbb{Z}$ and $\theta \in \Theta^{(g)}$, is called *neural process* and will be denoted by $\tilde{X}_\theta := (\tilde{X}_{t,\theta})_{t\in\mathbb{Z}}$.

Log-return Neural Process

The GAN objective for stochastic processes can be formulated as

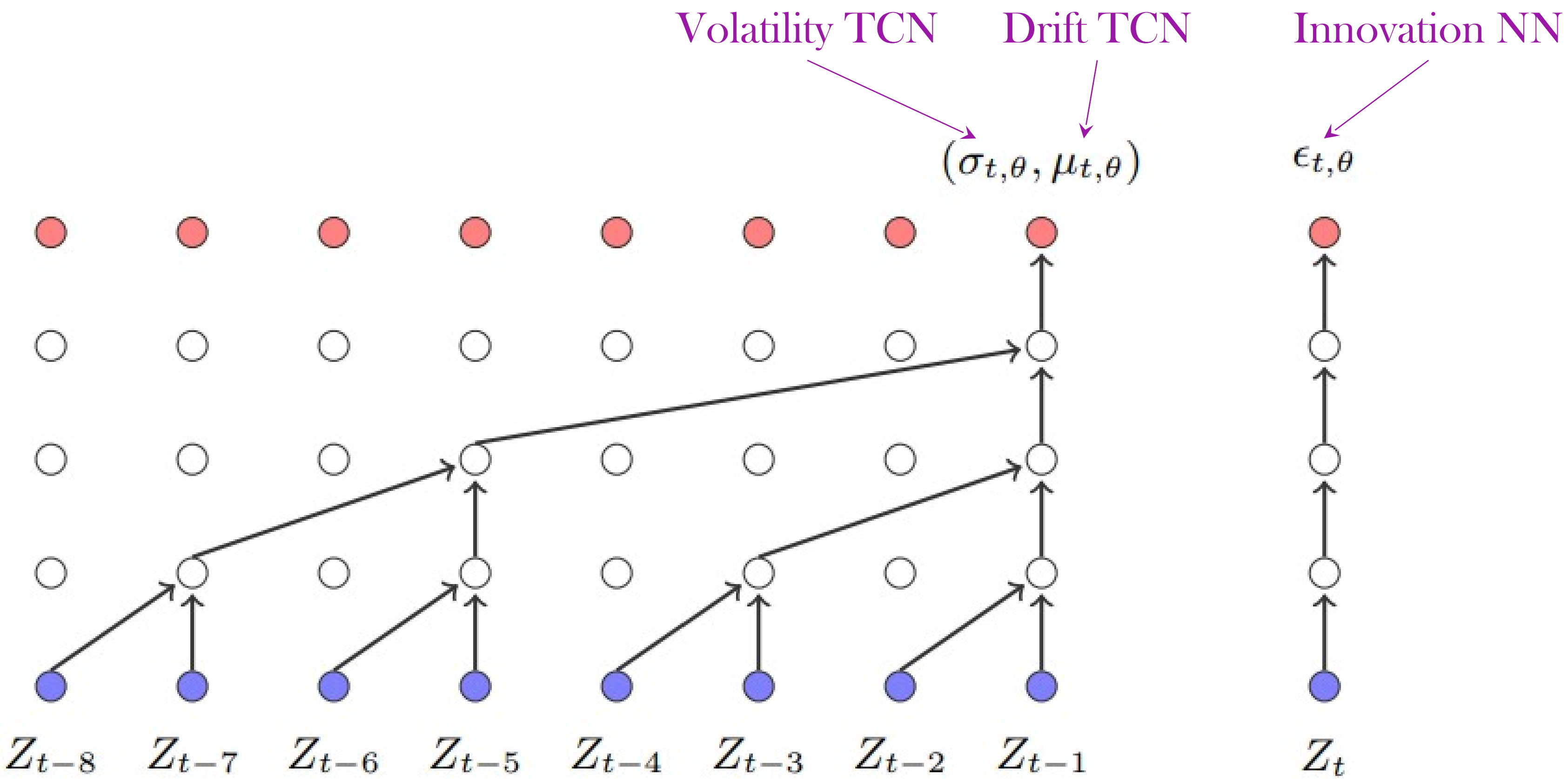
Monte Carlo estimate

where

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

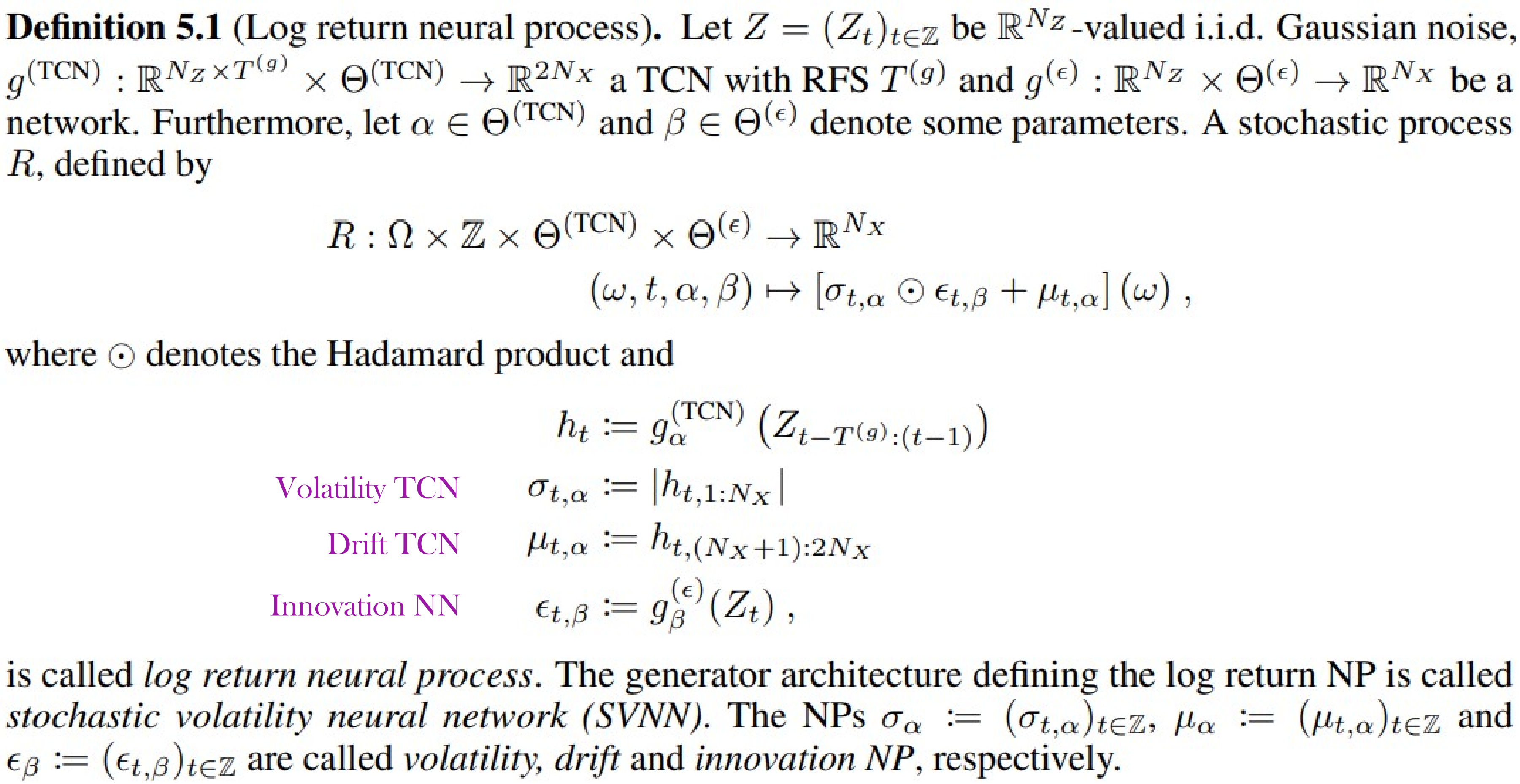
and and denote the real and the generated process, respectively.

Log-return Neural Process



Structure of the SVNN architecture. The volatility and drift component are generated by inferring the latent process through the TCN, whereas the innovation is generated by inferring .

Log-return Neural Process



Definition 5.1 (Log return neural process)**.** Let $Z = (Z_t)_{t\in\mathbb{Z}}$ be $\mathbb{R}^{N_Z}$-valued i.i.d. Gaussian noise, $g^{(\mathrm{TCN})} : \mathbb{R}^{N_Z \times T^{(g)}} \times \Theta^{(\mathrm{TCN})} \to \mathbb{R}^{2N_X}$ a TCN with RFS $T^{(g)}$ and $g^{(\epsilon)} : \mathbb{R}^{N_Z} \times \Theta^{(\epsilon)} \to \mathbb{R}^{N_X}$ be a network. Furthermore, let $\alpha \in \Theta^{(\mathrm{TCN})}$ and $\beta \in \Theta^{(\epsilon)}$ denote some parameters. A stochastic process R, defined by

$$\begin{aligned} R : \Omega \times \mathbb{Z} \times \Theta^{(\mathrm{TCN})} \times \Theta^{(\epsilon)} &\to \mathbb{R}^{N_X} \\ (\omega, t, \alpha, \beta) &\mapsto [\sigma_{t,\alpha} \odot \epsilon_{t,\beta} + \mu_{t,\alpha}](\omega)\ , \end{aligned}$$

where $\odot$ denotes the Hadamard product and

$$\begin{aligned} & h_t := g_\alpha^{(\mathrm{TCN})}\left(Z_{t-T^{(g)}:(t-1)}\right) \\ \text{Volatility TCN} \qquad & \sigma_{t,\alpha} := |h_{t,1:N_X}| \\ \text{Drift TCN} \qquad & \mu_{t,\alpha} := h_{t,(N_X+1):2N_X} \\ \text{Innovation NN} \qquad & \epsilon_{t,\beta} := g_\beta^{(\epsilon)}(Z_t)\ , \end{aligned}$$

is called *log return neural process*. The generator architecture defining the log return NP is called *stochastic volatility neural network (SVNN)*. The NPs $\sigma_\alpha := (\sigma_{t,\alpha})_{t\in\mathbb{Z}}$, $\mu_\alpha := (\mu_{t,\alpha})_{t\in\mathbb{Z}}$ and $\epsilon_\beta := (\epsilon_{t,\beta})_{t\in\mathbb{Z}}$ are called *volatility, drift* and *innovation NP*, respectively.

Table of Contents

I. Introduction

II. Generative Adversarial Networks (GANs)

III. Temporal Convolutional Networks (TCNs)

IV. Log-return Neural Process

V. Numerical Results

Numerical Results

QuantGANs Using Pure TCN

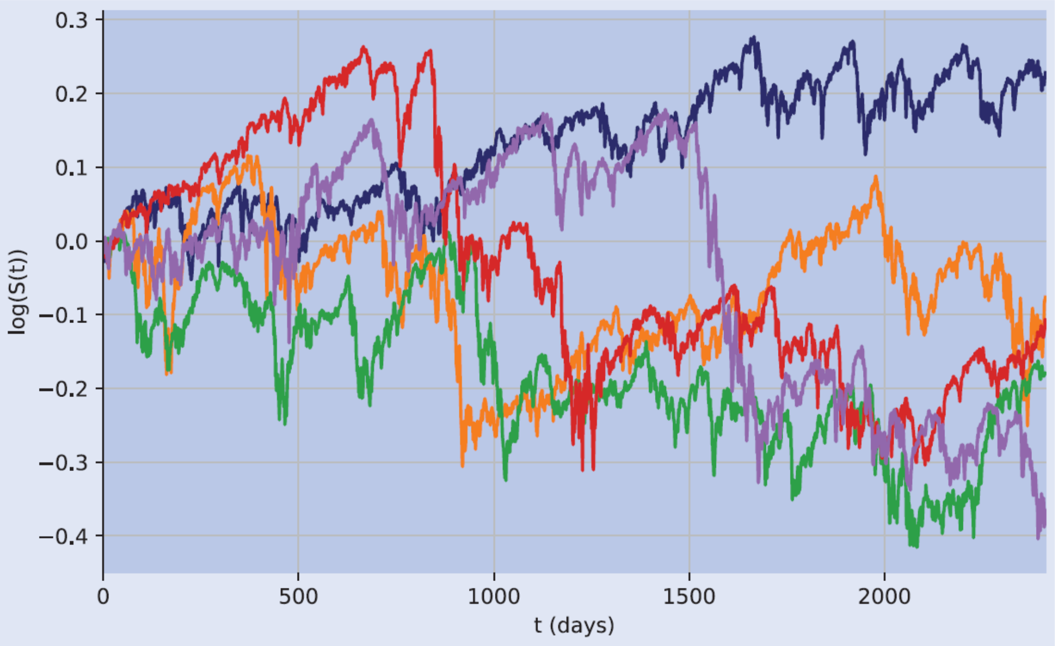


Figure A1. Five generated driftless log paths.

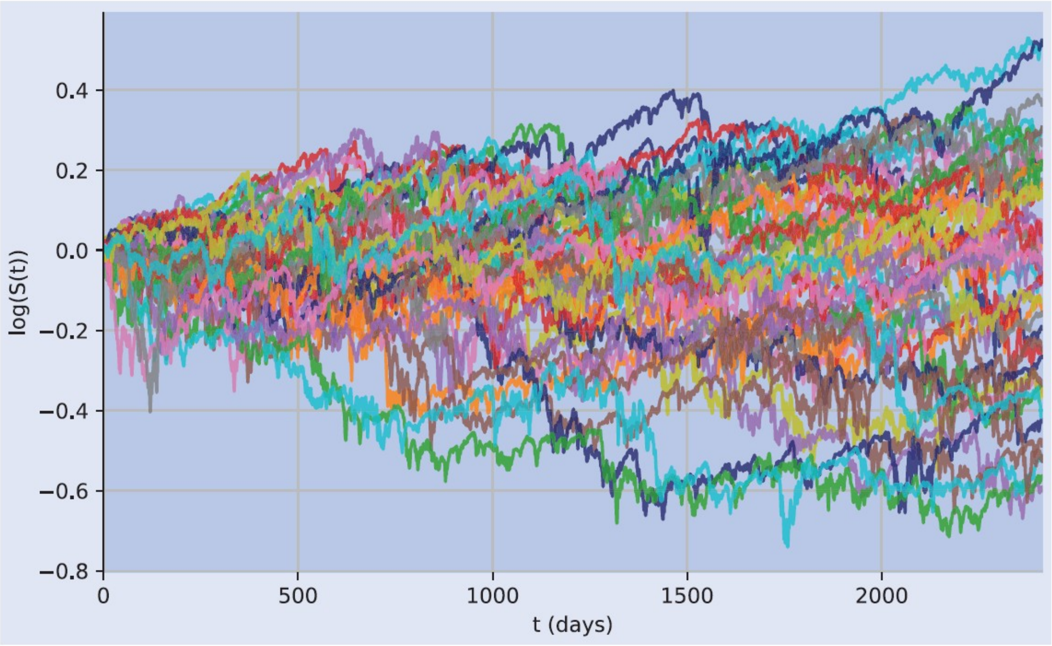
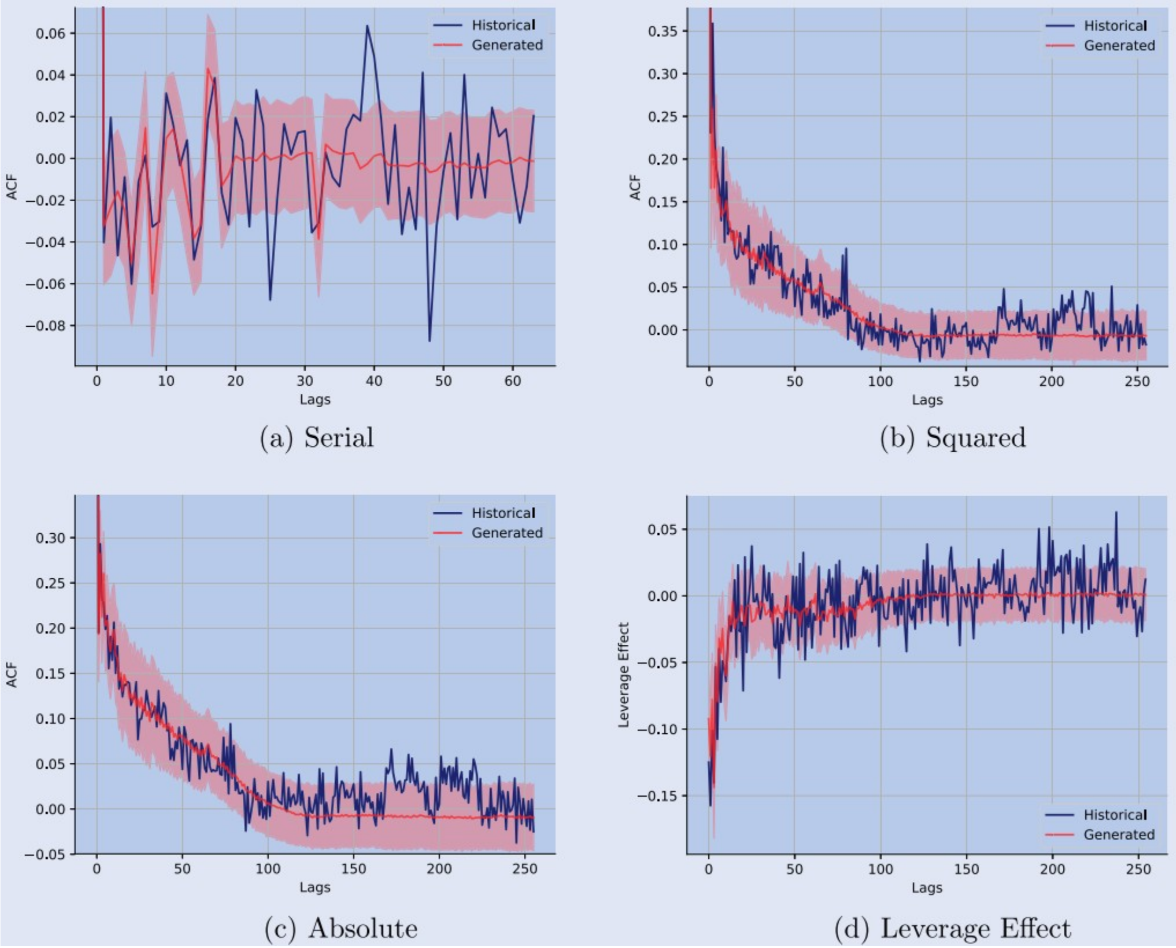


Figure A2. Fifty generated driftless log paths.

QuantGANs VS GARCH(1,1) (old model)

QuantGANs



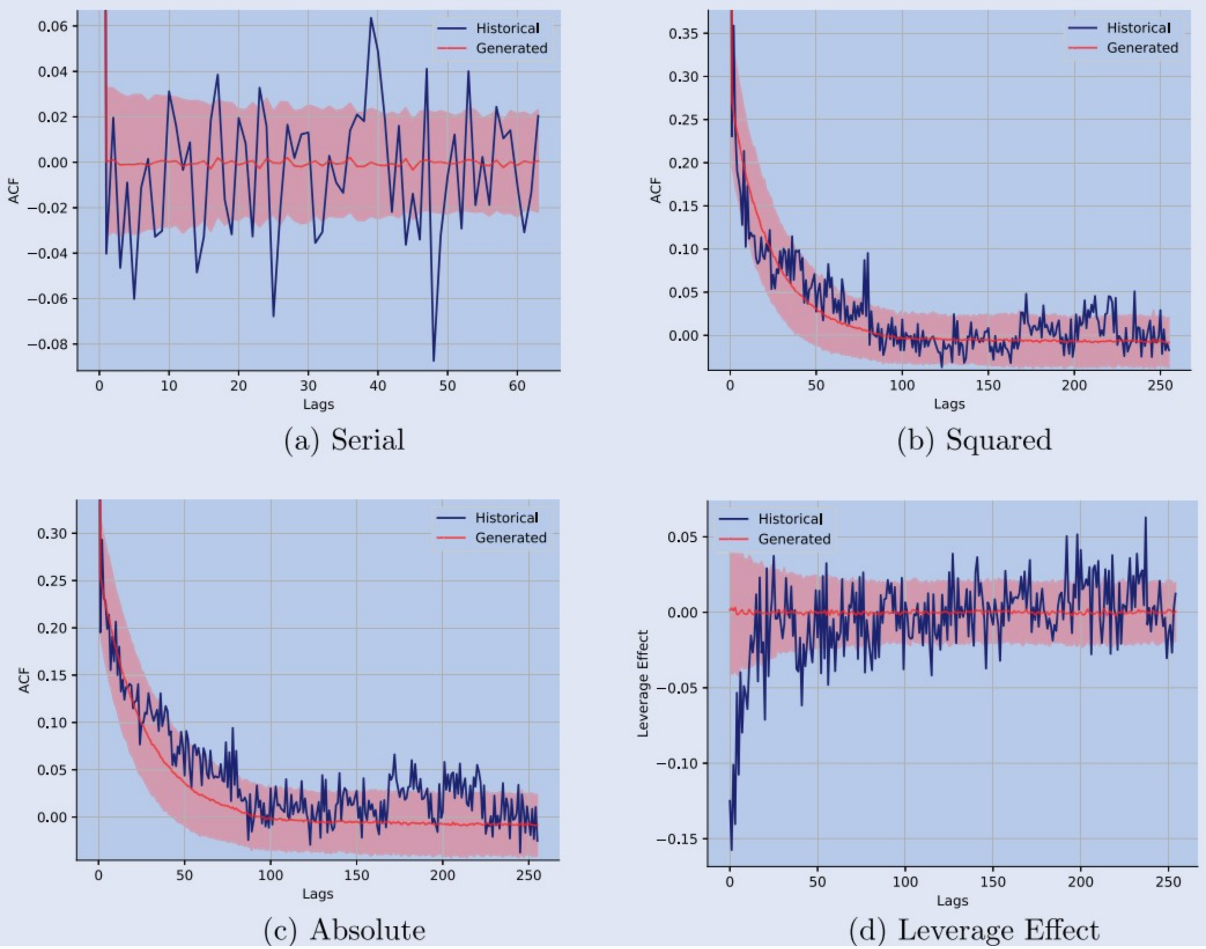
(a) Serial

(b) Squared

(c) Absolute

(d) Leverage Effect

GARCH(1,1) (old model)



(a) Serial

(b) Squared

(c) Absolute

(d) Leverage Effect

Thank you for listening