Quant GANs: Deep Generation of Financial Time Series

Hyelin Choi

Department of Mathematics

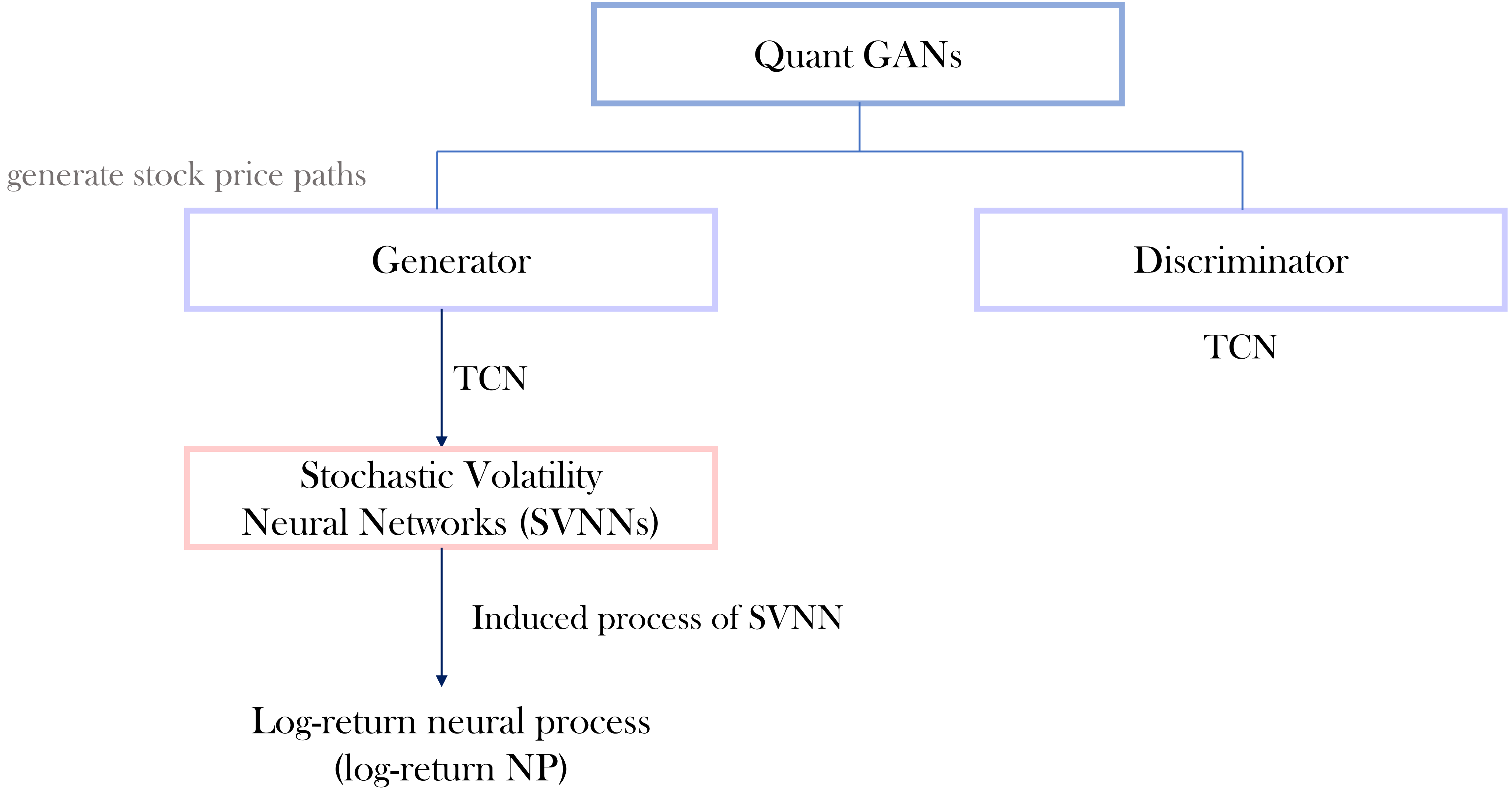
Sungkyunkwan University

April 15, 2024

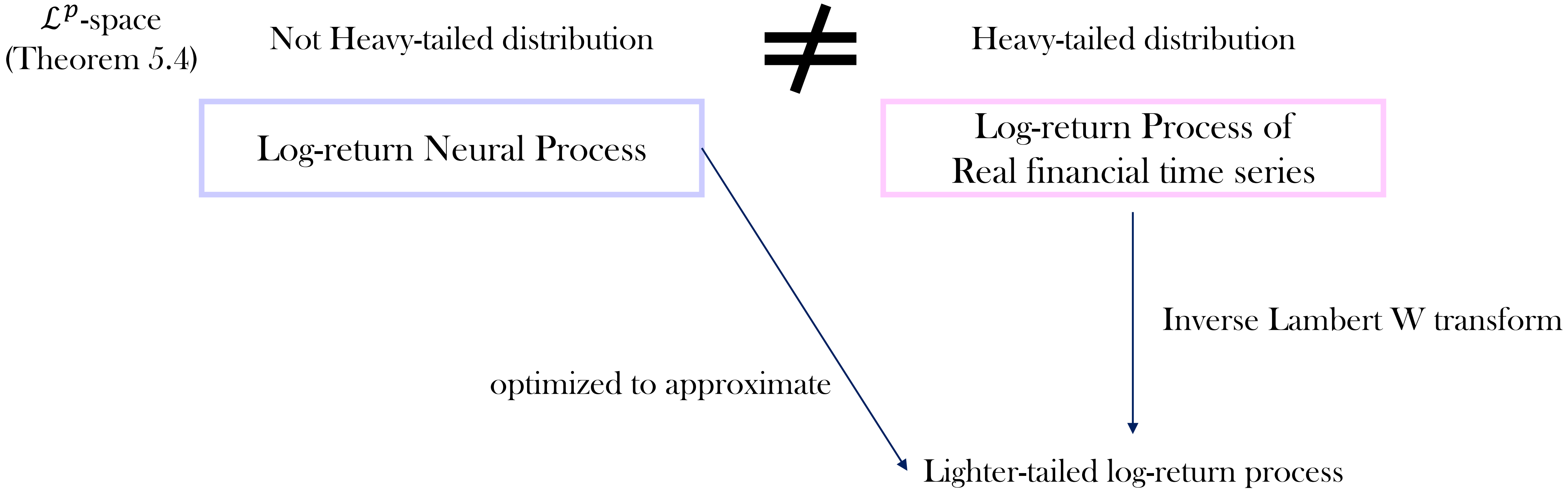
Table of Contents

Table of Contents

Overview


Quant GANs
generate stock price paths
Generator
Discriminator
TCN
TCN
Stochastic Volatility
Neural Networks (SVNNs)
Induced process of SVNN
Log-return neural process
(log-return NP)

Overview


$\mathcal{L}^p$-space
(Theorem 5.4)
Not Heavy-tailed distribution
≠
Heavy-tailed distribution
Log-return Neural Process
Log-return Process of
Real financial time series
Inverse Lambert W transform
optimized to approximate
Lighter-tailed log-return process

Overview

Can the risk-neutral distribution of log-return NP be derived?

In order to value options under a log-return NP, we should know a transition to its risk-neutral distribution.

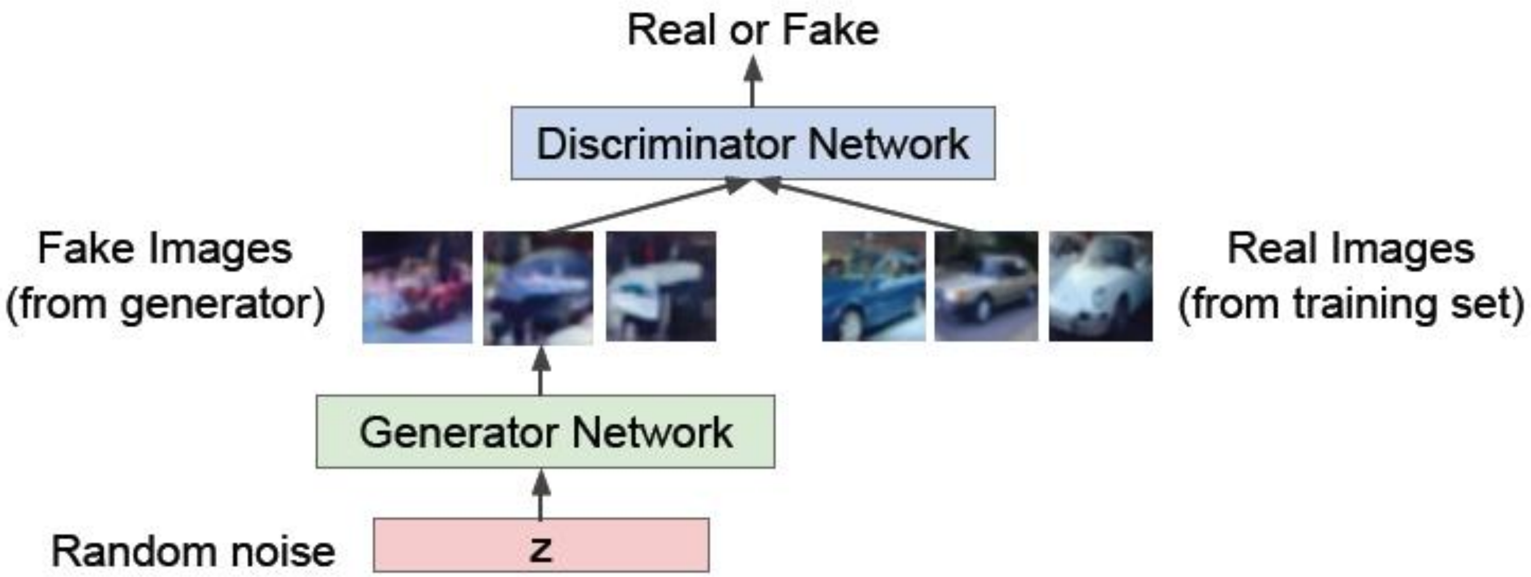
Table of Contents

Table of Contents

Generative Adversarial Networks

Remark 3.4. We call a function $f : \mathbb{R}^{d_0} \times \Theta \to \mathbb{R}^{d_1}$ with parameter space Θ a *network*, if it is Lipschitz continuous.

Generative Adversarial Networks



We will them to discrete-time stochastic process with TCNs.

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

Generative Adversarial Networks

- $(\mathbb{R}^{N_z}, \mathcal{B}(\mathbb{R}^{N_z}))$ and $(\mathbb{R}^{N_X}, \mathcal{B}(\mathbb{R}^{N_X}))$ are the latent and the data measure space, respectively.

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 $g\colon \mathbb{R}^{N_z} \times \Theta^{(g)} \to \mathbb{R}^{N_x}$ such that the induced random variable $g_\theta(Z) \coloneqq g_\theta \circ Z$ for some parameter $\theta \in \Theta^{(g)}$ and the targeted random variable X have the same distribution, i.e. $g_\theta(Z) \stackrel{\text{d}}{=} X$.

Generative Adversarial Networks

Definition 4.1 (Generator). Let $g : \mathbb{R}^{N_Z} \times \Theta^{(g)} \to \mathbb{R}^{N_X}$ be a network with parameter space $\Theta^{(g)}$. The random variable $\tilde{X}$, defined by

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

is called the *generated random variable*. Furthermore, the network g is called *generator* and $\tilde{X}_\theta$ the *generated random variable with parameter* θ.[3]

Generative Adversarial Networks

Definition 4.2 (Discriminator)**.** Let $\tilde{d} : \mathbb{R}^{N_X} \times \Theta^{(d)} \to \mathbb{R}$ be a network with parameters $\eta \in \Theta^{(d)}$ and $\sigma : \mathbb{R} \to [0,1] : x \mapsto \frac{1}{1+e^{-x}}$ be the sigmoid function. A function $d : \mathbb{R}^{N_X} \times \Theta^{(d)} \to [0,1]$ defined by $d : (x, \eta) \mapsto \sigma \circ \tilde{d}_\eta(x)$ is called a *discriminator*.

Generative Adversarial Networks

Definition 4.3 (Sample). A collection $\{Y_i\}_{i=1}^M$ of M independent copies of some random variable Y is called Y-*sample* of size M. The notation $\{y_i\}_{i=1}^M$ refers to a realisation $\{Y_i(\omega)\}_{i=1}^M$ for some $\omega \in \Omega$.

Generative Adversarial Networks

Loss function of GANs

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

Step 1

The discriminator's parameter $\eta \in \Theta^{(d)}$ are chosen to maximize the function $\mathcal{L}(\theta, \cdot), \theta \in \Theta^{(g)}$.

Generative Adversarial Networks

Loss function of GANs

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

Step 2

The generator's parameters $\theta \in \Theta^{(g)}$ are trained to minimized the probability of generated samples being identified as such and not from the data distribution.

Generative Adversarial Networks

We receive the min-max game

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

which refer to as the GAN objective.

Generative Adversarial Networks

Algorithm 1 GAN optimization.

INPUT: generator g, discriminator d, sample size $M \in \mathbb{N}$, generator learning rate α_g, discriminator learning rate α_d, number of discriminator optimization steps k
OUTPUT: parameters (θ, η)

while not converged **do**
  for k steps **do**
    Let $\{\tilde{x}_{\theta,i}\}_{i=1}^{M}$ be a realisation of an $\tilde{X}_\theta$-sample of size M.
    Let $\{x_i\}_{i=1}^{M}$ be a realisation of an X-sample of size M.
    Compute and store the gradient

$$\Delta_\eta \leftarrow \nabla_\eta \frac{1}{M} \sum_{i=1}^{M} \log(d(x_i)) + \log(1 - d(\tilde{x}_{\theta,i})) \ .$$

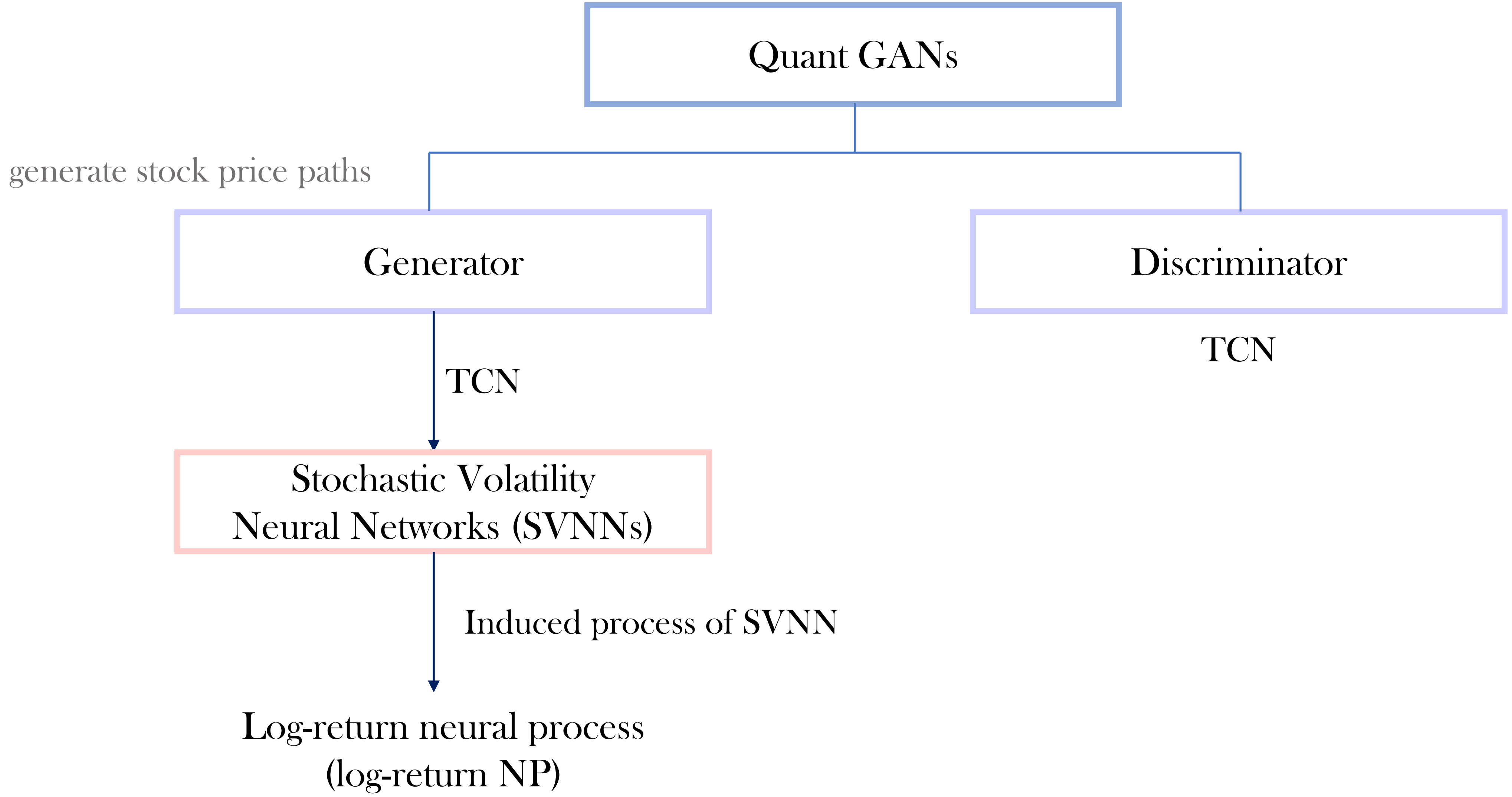
    Ascent the discriminator's parameters: $\eta \leftarrow \eta + \alpha_d \cdot \Delta_\eta$.
  end for
  Let $\{\tilde{x}_{\theta,i}\}_{i=1}^{M}$ be a realisation of an $\tilde{X}_\theta$-sample of size M.
  Compute and store the gradient

$$\Delta_\theta \leftarrow \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log(d(\tilde{x}_{\theta,i})) \ .$$

  Descent the generator's parameters: $\theta \leftarrow \theta - \alpha_g \cdot \Delta_\theta$.
end while

Table of Contents

Overview


Quant GANs
generate stock price paths
Generator
Discriminator
TCN
TCN
Stochastic Volatility
Neural Networks (SVNNs)
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

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

and for an ω-realization

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

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 $\tilde{X}_\theta := (\tilde{X}_{t,\theta})_{t\in\mathbb{Z}}$.

In the context of GANs, the i.i.d. noise process $Z = (Z_t)_{t\in\mathbb{Z}}$ from Definition 4.5 represents the noise prior.

We assume that for all $t \in \mathbb{Z}$ the random variable Z_t follows a multivariate standard normal distribution, i.e.

$Z_t \sim N(0, I)$

Log-return Neural Process

The GAN objective for stochastic processes can be formulated as

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

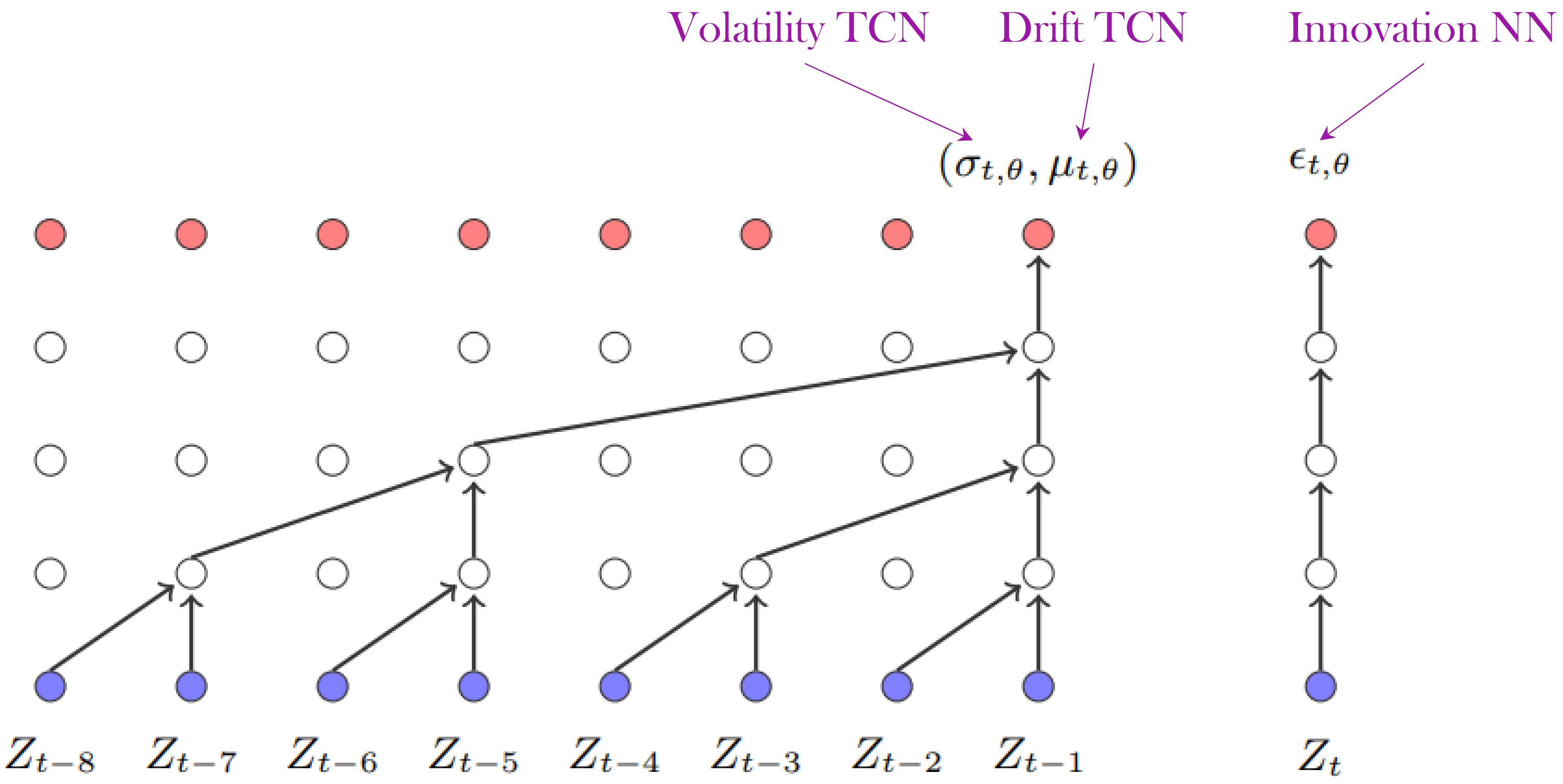
Monte Carlo estimate

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 $X_{1:T^{(d)}}$ and $\tilde{X}_{1:T^{(d)},\theta}$ denote the real and the generated process, respectively.

Log-return Neural Process



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

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

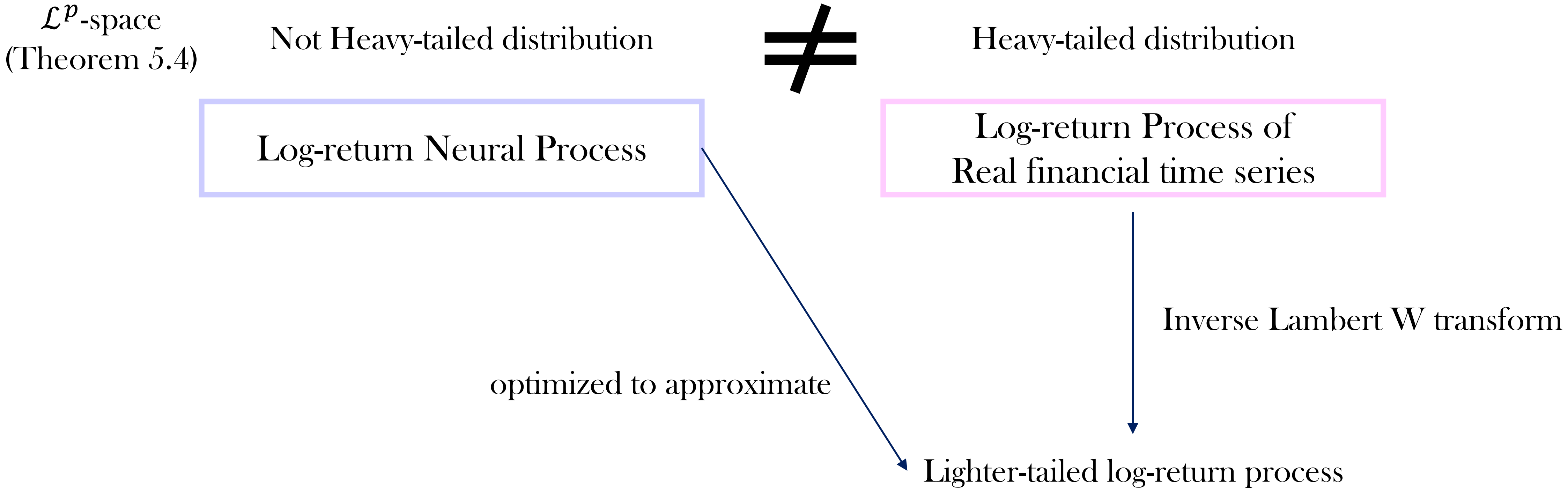
where $\odot$ denotes the Hadamard product and

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

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

Overview


$\mathcal{L}^p$-space
(Theorem 5.4)
Not Heavy-tailed distribution
≠
Heavy-tailed distribution
Log-return Neural Process
Log-return Process of
Real financial time series
Inverse Lambert W transform
optimized to approximate
Lighter-tailed log-return process

L^p-space Characterization of R_θ

Theorem 5.4 (L^p-characterization of neural networks). *Let* $p \in \mathbb{N}$, $Z \in L^p(\mathbb{R}^{N_Z})$ *and* $g : \mathbb{R}^{N_Z} \times \Theta \to \mathbb{R}^{N_X}$ *a network with parameters* $\theta \in \Theta$. *Then,* $g_\theta(Z) \in L^p(\mathbb{R}^{N_X})$.

$$h_t := g_\alpha^{(\mathrm{TCN})}\left(Z_{t-T^{(g)}:(t-1)}\right)$$

Volatility TCN $\quad \sigma_{t,\alpha} := \left|h_{t,1:N_X}\right|$

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

Innovation NN $\quad \epsilon_{t,\beta} := g_\beta^{(\epsilon)}(Z_t)\ ,$

$$\sigma_{t,\theta}, \varepsilon_{t,\theta}, \mu_{t,\theta} \in L^p(\mathbb{R}^{N_X})$$

L^p-space Characterization of R_θ

Corollary 5.5. *Let* R_θ *be a log return NP parametrized by some* $\theta \in \Theta$. *Then, for all* $t \in \mathbb{Z}$ *and* $p \in \mathbb{N}$ *the random variable* $R_{t,\theta}$ *is an element of the space* $L^p(\mathbb{R}^{N_X})$.

↓

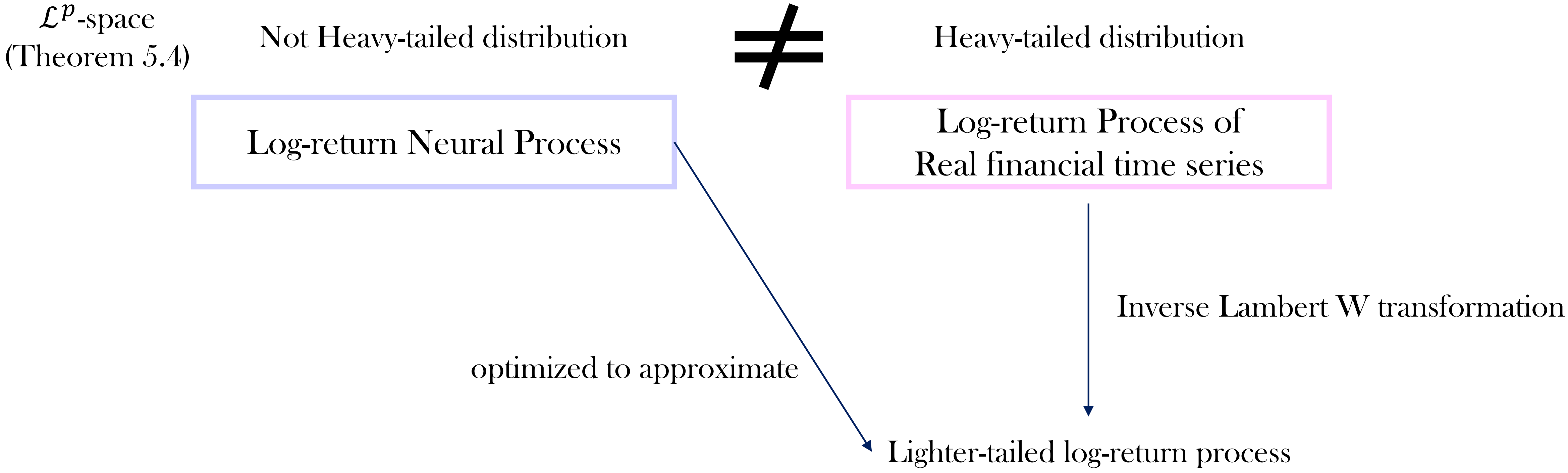
All moments of the log-return NP exist.

↓

The log-returns NP does not exhibit heavy tail.

Table of Contents

Inverse Lambert W Transform



Inverse Lambert W Transform

Definition 5.7 (Lambert W$\times F_X$). Let $\delta \in \mathbb{R}$ and X be an $\mathbb{R}$-valued random variable with mean μ, standard deviation σ and cumulative distribution function F_X. The location-scale Lambert W$\times F_X$ transformed random variable Y is defined by

$$Y = U \exp\left(\frac{\delta}{2}U^2\right)\sigma + \mu\,, \tag{3}$$

where $U := \dfrac{X-\mu}{\sigma}$ is the normalizing transform.

Bijective
&
differentiable

The Lambert W probability transform is used to generate heavier tails.

Inverse Lambert W Transform

Lambert W function is the inverse of $z = u\, exp(u)$, that is, that function which satisfies $W(z)\exp\big(W(z)\big) = z$.

Inverse Lambert W Transform

The inverse Lambert W transformation is

$$W(Y) := W_\delta\left(\frac{Y-\mu}{\sigma}\right)\sigma + \mu$$

where

$$W_\delta(z) := \operatorname{sgn}(z)\left(\frac{W(\delta z^2)}{\delta}\right)^{\frac{1}{2}},$$

and $\operatorname{sgn}(z)$ is the sign of z and . $W_\delta(z)$ is bijective for all $\delta \geq 0$ and all $z \in \mathbb{R}$.

The model parameter can be estimated via quasi maximum likelihood.

Inverse Lambert W Transform

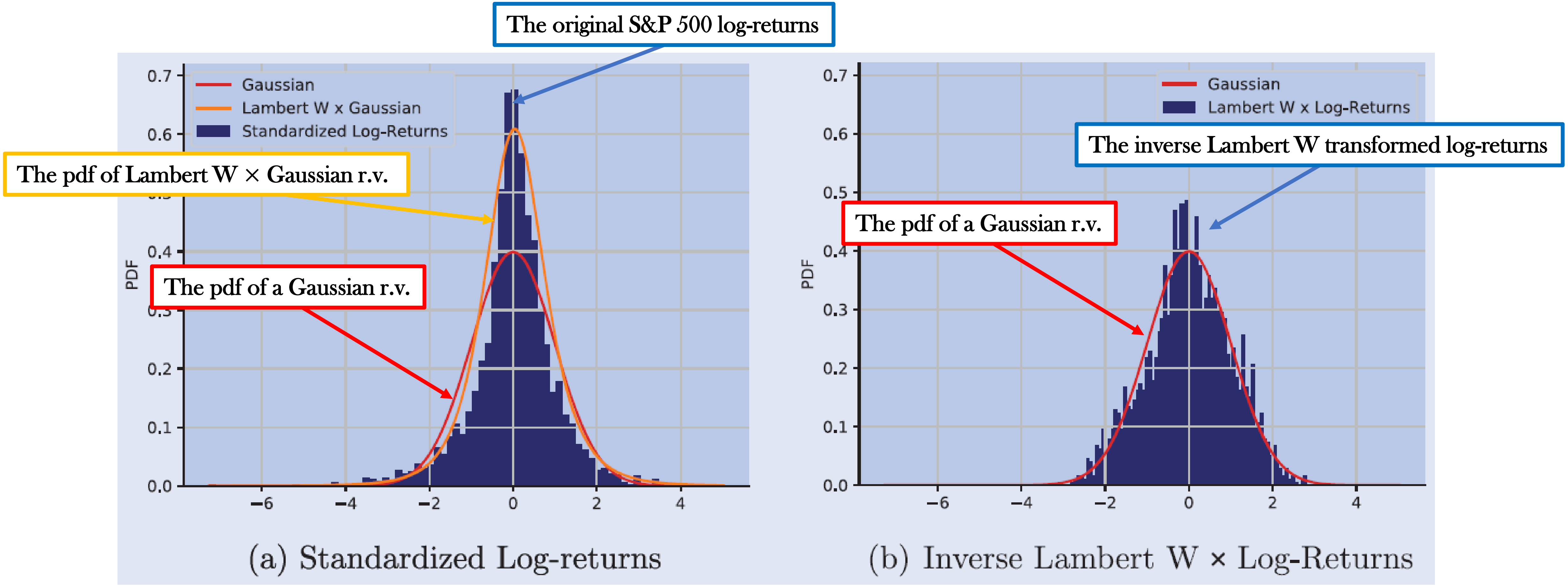


Figure 10. (a) The original S&P 500 log-returns and the fitted probability density function of a Lambert W × Gaussian random variable. (b) The inverse Lambert W transformed log-returns and the probability density function of a Gaussian random variable.

Overview

$\mathcal{L}^p$-space (Theorem 5.4)

Not Heavy-tailed distribution $\neq$ Heavy-tailed distribution

Log-return Neural Process

Log-return Process of real financial time series

Inverse Lambert W transformation

optimized to approximate

Lighter-tailed log-return process

$R^W := (R_t^W)_{t\in\mathbb{N}}$

Inverse Lambert W Transform

Assumption 1. The inverse Lambert W transformed spot log returns R^W can be represented by a log return neural process R_θ for some $\theta \in \Theta$.

<u>Implications of Assumption 1</u>

1. The log return NP is stationary such that the historical log return process is assumed to be stationary, by construction.
2. Log-return NPs can capture dynamics up to the RFS of the TCN in use. Therefore, Assumption 1 implies for an RFS $T^{(g)}$ that for any $t \in \mathbb{Z}$ the random variables $R_t, R_{t+T^{(g)}+1}$ are independent.

Table of Contents

Risk-neutral Log-return NP

Can the risk-neutral distribution of log-return NP be derived?

In order to value options under a log-return NP, we should know a transition to its risk-neutral distribution.

Risk-neutral Log-return NP

- One-dimensional log-return NP $$R_{t,\theta} = \sigma_{t,\theta}\,\epsilon_{t,\theta} + \mu_{t,\theta}$$

- Spot price $$S_{t,\theta} = S_{t-1,\theta}\,\exp(R_{t,\theta})$$

$$r_t = \log\left(\frac{s_t}{s_{t-1}}\right)$$

- Discounted Spot price $$\tilde{S}_{t,\theta} := \frac{S_{t,\theta}}{\exp(rt)}$$

$$\tilde{S}_{t,\theta} = \tilde{S}_{t-1,\theta}\,\exp(R_{t,\theta} - r)$$

Risk-neutral Log-return NP

In risk-neutral representation, the discounted stock price process has to be a martingale.

$$\mathbb{E}[\tilde{S}_{t,\theta}|\mathcal{F}^Z_{t-1}] = \mathbb{E}[\tilde{S}_{t-1,\theta}\ \exp(R_{t,\theta} - r)|\mathcal{F}^Z_{t-1}]$$

$\tilde{S}_{t-1,\theta}$ is F^Z_{t-1}-measurable →

$$= \tilde{S}_{t-1,\theta}\ \exp(-r)\,\mathbb{E}[\exp(\sigma_{t,\theta}\,\epsilon_{t,\theta} + \mu_{t,\theta})|\mathcal{F}^Z_{t-1}]$$

$\sigma_{t,\theta}$ and $\mu_{t,\theta}$ is F^Z_{t-1}-measurable and $\varepsilon_{t,\theta}$ is independent of F^Z_{t-1}

$$\mathbb{E}[\exp(\sigma_{t,\theta}\,\epsilon_{t,\theta} + \mu_{t,\theta})|\mathcal{F}^Z_{t-1}] = \mathbb{E}[\exp(\sigma\,\epsilon_{t,\theta} + \mu)]_{\substack{\sigma=\sigma_{t,\theta}\\ \mu=\mu_{t,\theta}}} =: h(\sigma_{t,\theta}, \mu_{t,\theta})$$

Risk-neutral Log-return Neural Process

$$R^M_{t,\theta} := R_{t,\theta} - \log(h(\sigma_{t,\theta}, \mu_{t,\theta})) + r$$

Risk-neutral Log-return NP

Discounted risk-neutral spot price process

$$\tilde{S}^M_{t,\theta} = \tilde{S}^M_{t-1,\theta}\ \exp(R^M_{t,\theta} - r) = \tilde{S}^M_{t-1,\theta}\ \exp(R_{t,\theta} - \log(h(\sigma_{t,\theta}, \mu_{t,\theta})))$$

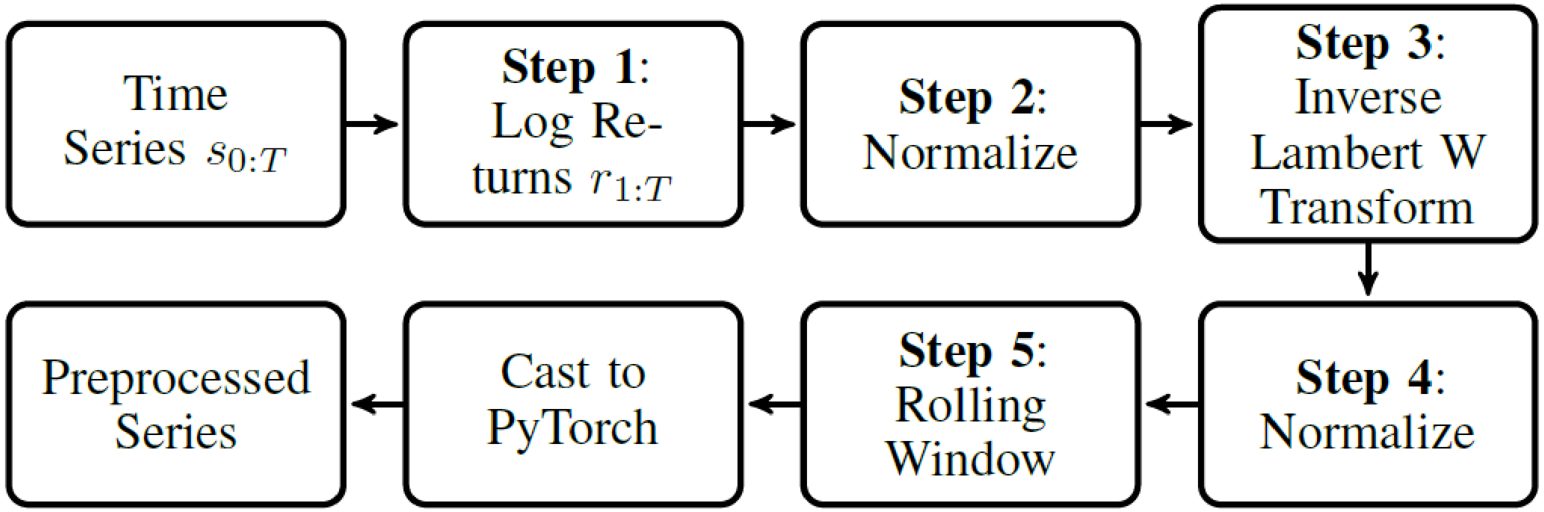
Explicit formula for the (discounted) risk-neutral spot price process

$$\tilde{S}^M_{t,\theta} = S_0\ \exp\left(\sum_{s=1}^{t}[R_{s,\theta} - \log(h(\sigma_{s,\theta}, \mu_{s,\theta}))]\right)$$

$$S^M_{t,\theta} = S_0\ \exp\left(\sum_{s=1}^{t}[R_{s,\theta} - \log(h(\sigma_{s,\theta}, \mu_{s,\theta}))] + rt\right)$$

Table of Contents

Preprocessing



Preprocessing

- *Step 1*: Log-returns $r_{1:T}$: Calculate the log-return series

$$r_t = \log\left(\frac{s_t}{s_{t-1}}\right) \qquad for\ all\ t \in \{1, \cdots, T\}.$$

Preprocessing

- *Step 2 & 4*: Normalize:

We normalize the data in order to obtain a series with zero mean and unit variance.

Preprocessing

- *Step 3*: Inverse Lambert W transform:

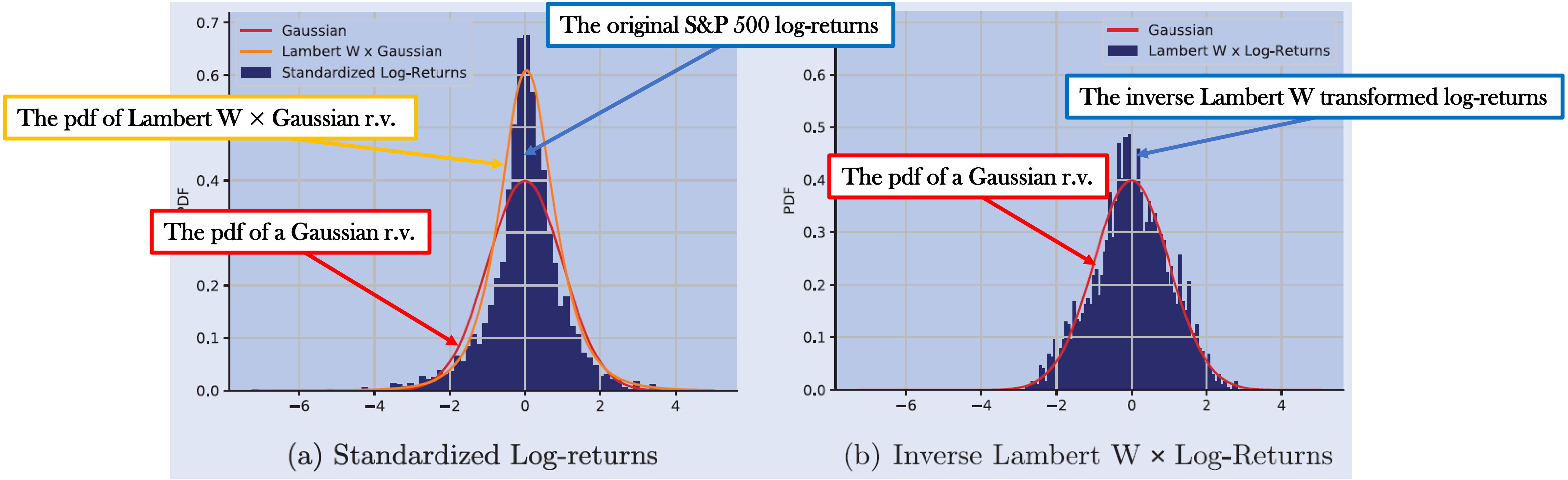


Figure 10. (a) The original S&P 500 log-returns and the fitted probability density function of a Lambert W × Gaussian random variable. (b) The inverse Lambert W transformed log-returns and the probability density function of a Gaussian random variable.

Preprocessing

- *Step 5*: Rolling window: When considering a discriminator with receptive field size $T^{(d)}$, we apply a rolling window of corresponding length and stride one to the preprocessed. log-return sequence $\gamma_t^{(\rho)}$ Hence, for $t \in \{1, \cdots, T - T^{(d)}\}$ we define the subsequences

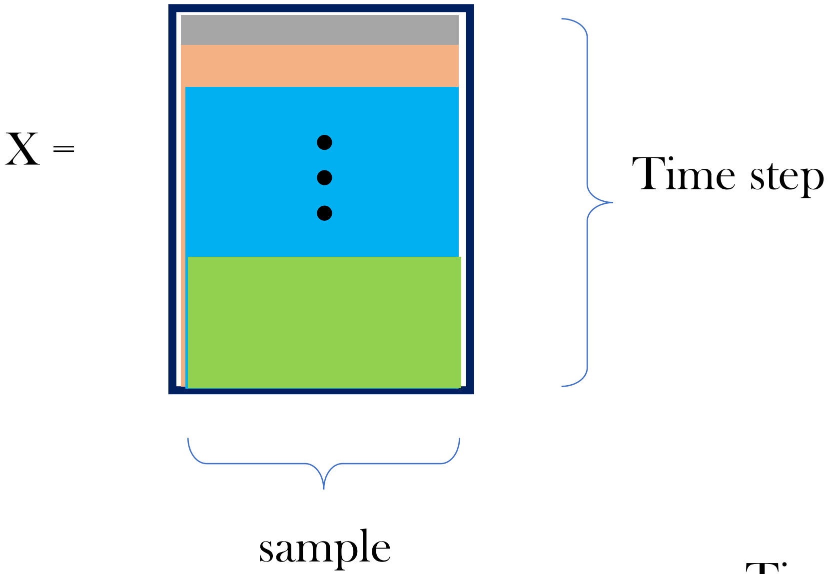
$$\gamma_{1:T^{(d)}}^{(t)} \coloneqq \gamma_{t:T^{(d)}+t-1}^{(\rho)} \in \mathbb{R}^{N_z \times T^{(d)}}.$$

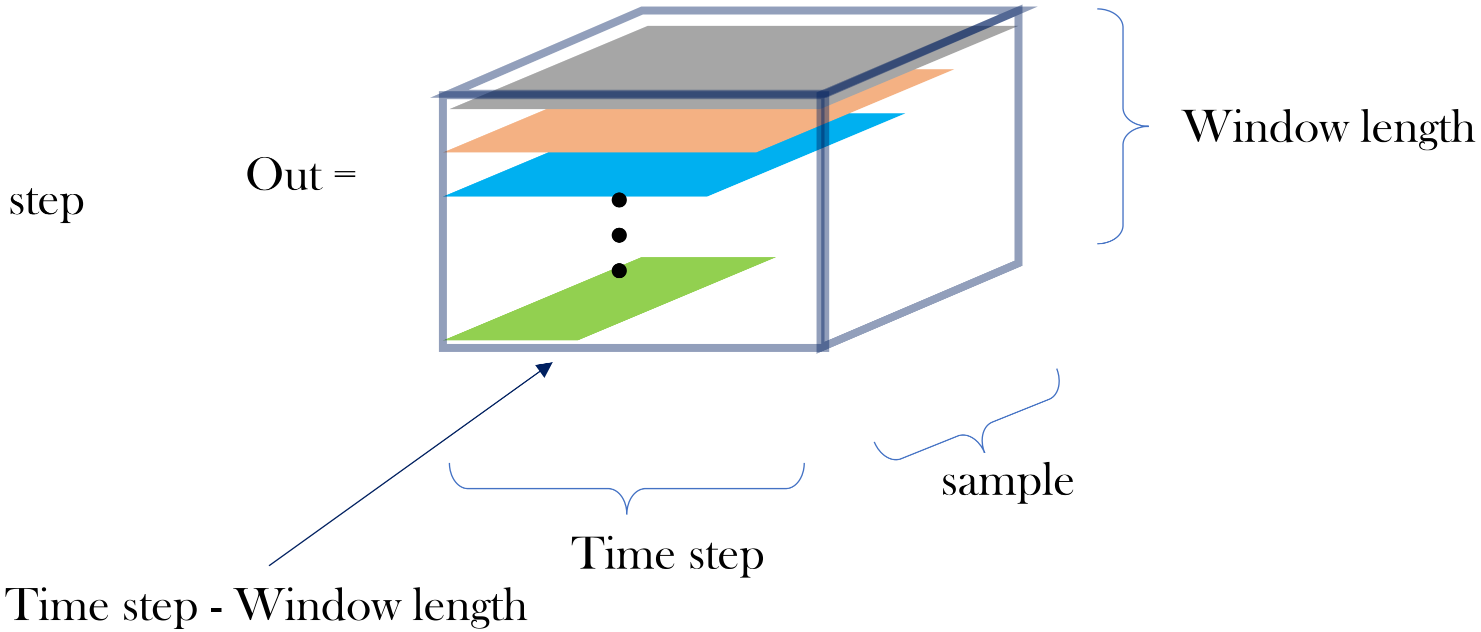
Preprocessing

```python
import numpy as np
from sklearn.preprocessing import StandardScaler
from backend.gaussianize import Gaussianize

def rolling_window(x, k, sparse=True):
    """compute rolling windows from timeseries

    Args:
        x ([2d array]): x contains the time series in the shape (timestep, sample).
        k ([int]): window length.
        sparse (bool): Cut off the final windows containing NA. Defaults to True.

    Returns:
        [3d array]: array of rolling windows in the shape (window, timestep, sample).
    """
    out = np.full([k, *x.shape], np.nan)
    N = len(x)
    for i in range(k):
        out[i, :N-i] = x[i:]

    if not sparse:
        return out

    return out[:, :-(k-1)]
```

Preprocessing

Time step (N)
Sample
Window length (k)

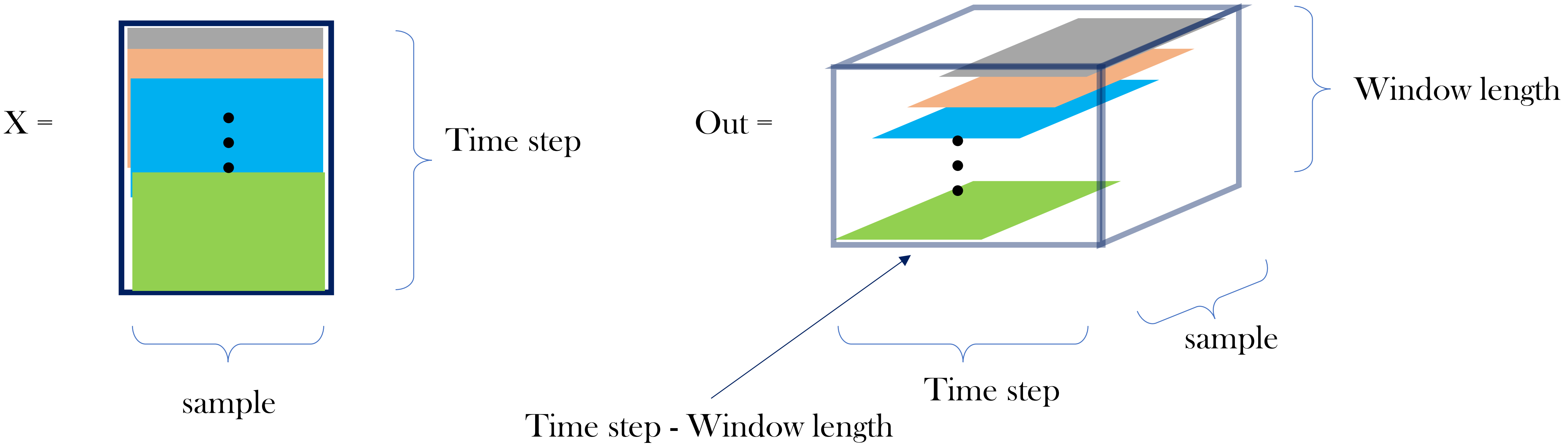




- Sparse = True
- Sparse = False

Preprocessing

Time step (N)
Sample
Window length (k)



- Sparse = True
- Sparse = False

Table of Contents

Numerical Results

We test the generative capabilities of Quant GANs by modeling the log-returns of the S&P 500 index.

1) Pure TCN
2) Constrained log-return NP
3) GARCH(1,1) with constant drift (for comparison)

We will show that Quant GANs can learn neural process that matches the empirical distribution and dependence properties fat better than the presented GARCH model.

Numerical Results

<u>Constraints</u>

1. One-dimensional case ($N_X = 1$)
2. The innovation NP represents a standard normal distributed random variable

$$\varepsilon_{t,\theta} \sim N(0,1) \quad for\ all\ t \in \mathbb{Z}\,.$$

$h(\sigma_{t,\theta}, \mu_{t,\theta})$ can be calculated explicitly:

$$h(\sigma_{t,\theta}, \mu_{t,\theta}) = \mathbb{E}[\exp(\underbrace{\sigma\, \epsilon_{t,\theta} + \mu}_{\sim \mathcal{N}(\mu, \sigma^2)})]_{\substack{\sigma=\sigma_{t,\theta} \\ \mu=\mu_{t,\theta}}} = \exp\left(\mu_{t,\theta} + \frac{\sigma_{t,\theta}^2}{2}\right)$$

Risk-neutral Log-return Neural Process

$$R_{t,\theta}^M = \sigma_{t,\theta}\, \epsilon_{t,\theta} - \frac{\sigma_{t,\theta}^2}{2} + r$$

Numerical Results

<u>Constraints</u>

1. One-dimensional case ($N_X = 1$)
2. The innovation NP represents a standard normal distributed random variable

$$\varepsilon_{t,\theta} \sim N(0,1) \quad for\ all\ t \in \mathbb{Z}\,.$$

Discounted risk-neutral spot price process

$$\tilde{S}^M_{t,\theta} = \tilde{S}^M_{t-1,\theta} \exp\left(\sigma_{t,\theta}\,\epsilon_{t,\theta} - \frac{\sigma^2_{t,\theta}}{2}\right)$$

Explicit formula for the (discounted) risk-neutral spot price process

$$\tilde{S}^M_{t,\theta} = S_0 \exp\left(\sum_{s=1}^{t}\left(\sigma_{s,\theta}\,\epsilon_{s,\theta} - \frac{\sigma^2_{s,\theta}}{2}\right)\right)$$

$$S^M_{t,\theta} = S_0 \exp\left(\sum_{s=1}^{t}\left[\sigma_{s,\theta}\,\epsilon_{s,\theta} - \frac{\sigma^2_{s,\theta}}{2}\right] + rt\right)$$

Numerical Results

- Evaluating a path simulator: metrics and scores

1. Distributional metrics
    1) Earth mover distance
    2) DY metric
2. Dependence scores
    1) ACF score
    2) Leverage effect score

Numerical Results

Earth mover distance

DY metric

ACF score

Leverage effect

Table 2. Evaluated metrics for the three models applied. For each row, the best value is printed bold.

	TCN	C-SVNN with drift	GARCH(1,1)
EMD(1)	**0.0039**	0.0040	0.0199
EMD(5)	**0.0039**	0.0040	0.0145
EMD(20)	**0.0040**	0.0069	0.0276
EMD(100)	**0.0154**	0.0464	0.0935
DY(1)	**19.1199**	19.8523	32.7090
DY(5)	**21.1167**	21.2445	27.4760
DY(20)	26.3294	**25.0464**	39.3796
DY(100)	28.1315	**25.8081**	46.4779
ACF(id)	**0.3363**	0.3482	0.3532
ACF($\|\cdot\|$)	**0.3823**	0.4549	0.4610
ACF($(\cdot)^2$)	**0.3398**	0.3878	0.4008
Leverage Effect	**0.3291**	0.3351	0.4636

Numerical Results

Pure TCN

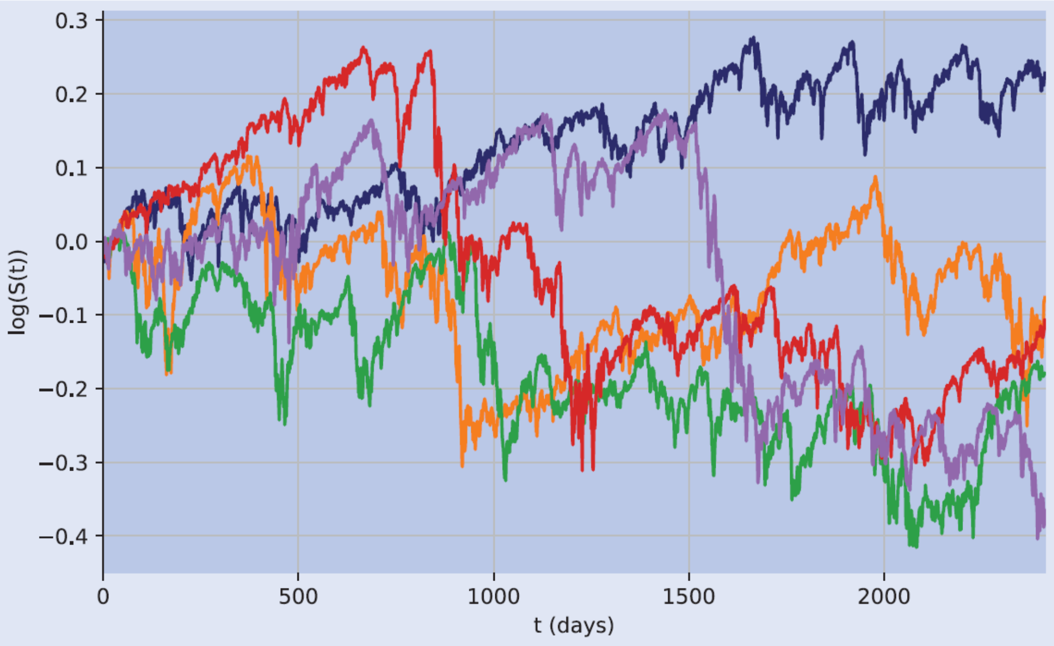


Figure A1. Five generated driftless log paths.

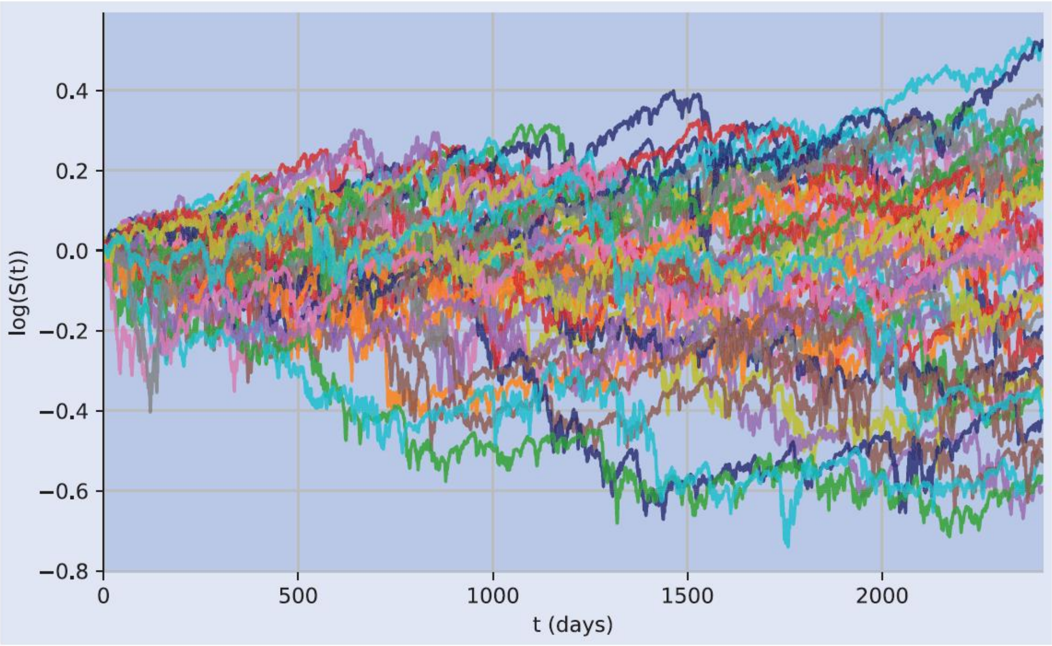


Figure A2. Fifty generated driftless log paths.

Numerical Results

Constrained SVNN

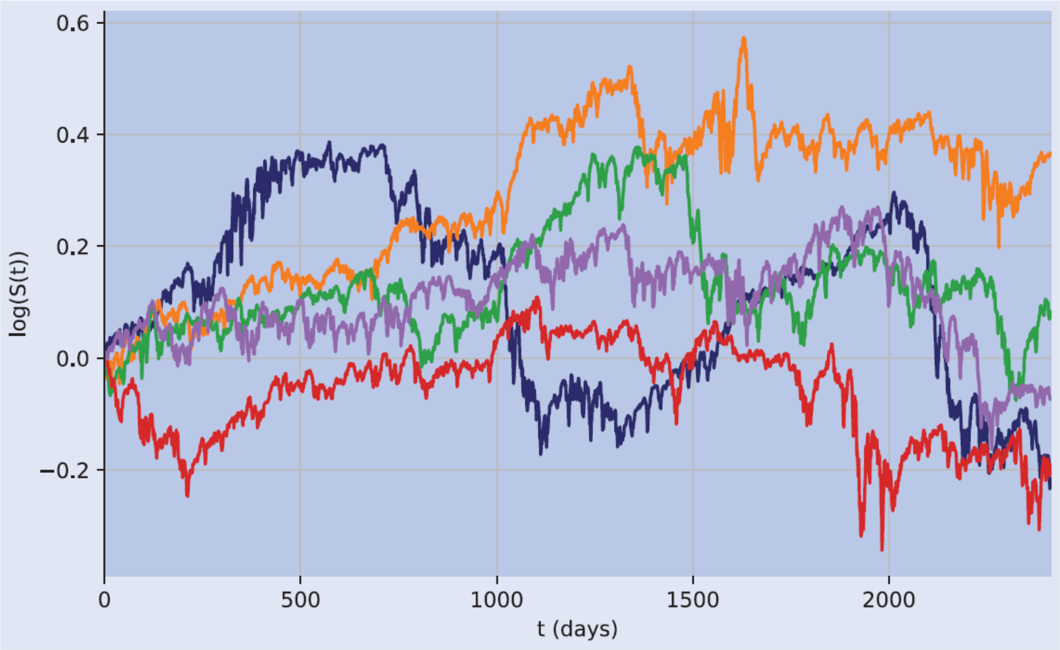


Figure A5. Five generated driftless log paths.

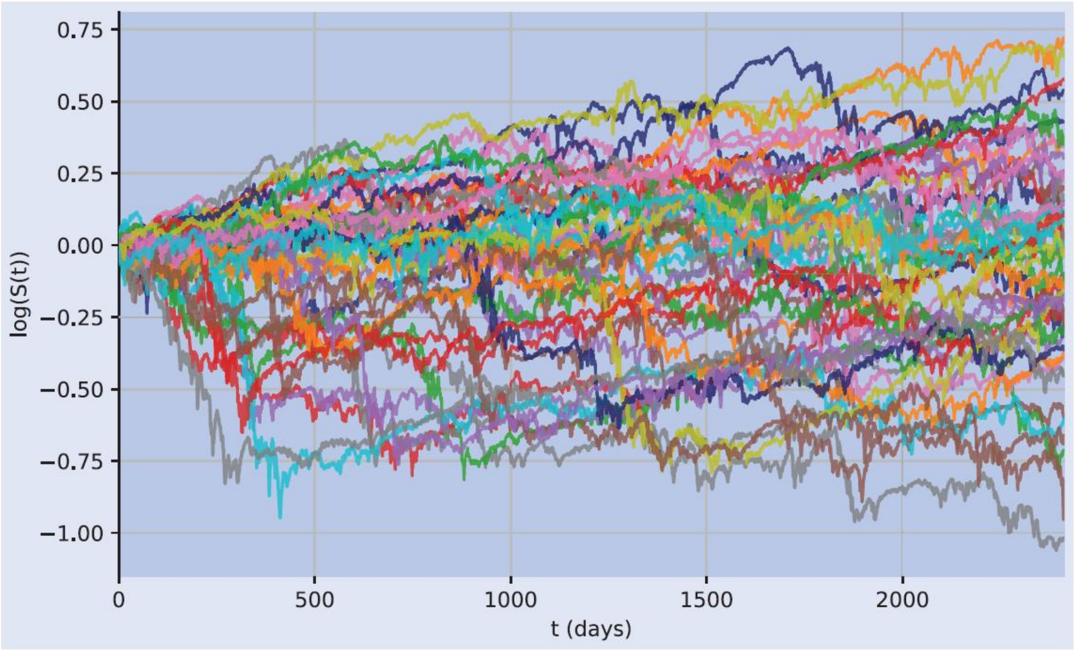


Figure A6. Fifty generated driftless log paths.

Numerical Results

GARCH(1,1) with constant drift

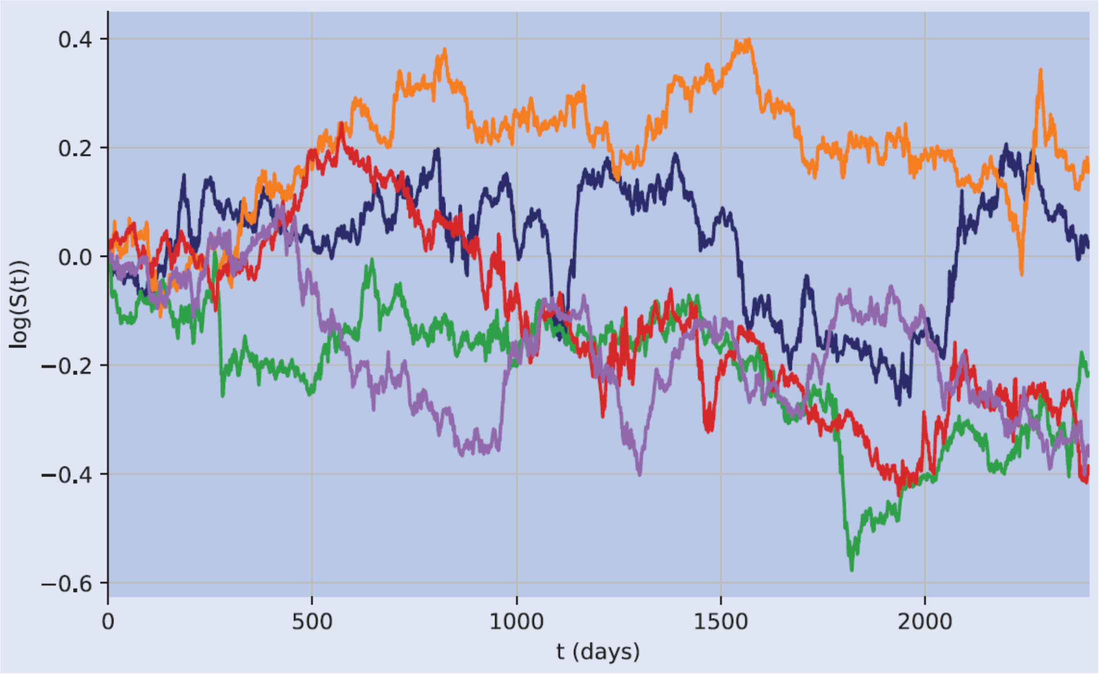


Figure A9. Five generated driftless log paths.

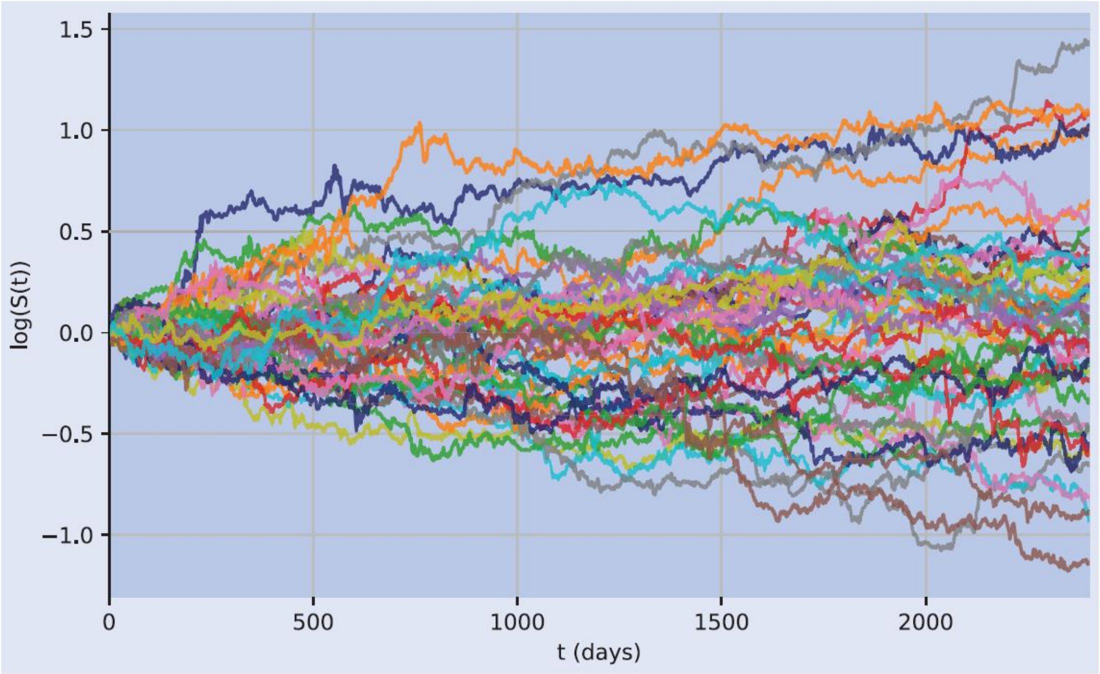


Figure A10. Fifty generated driftless log paths.

Numerical Results

Pure TCN

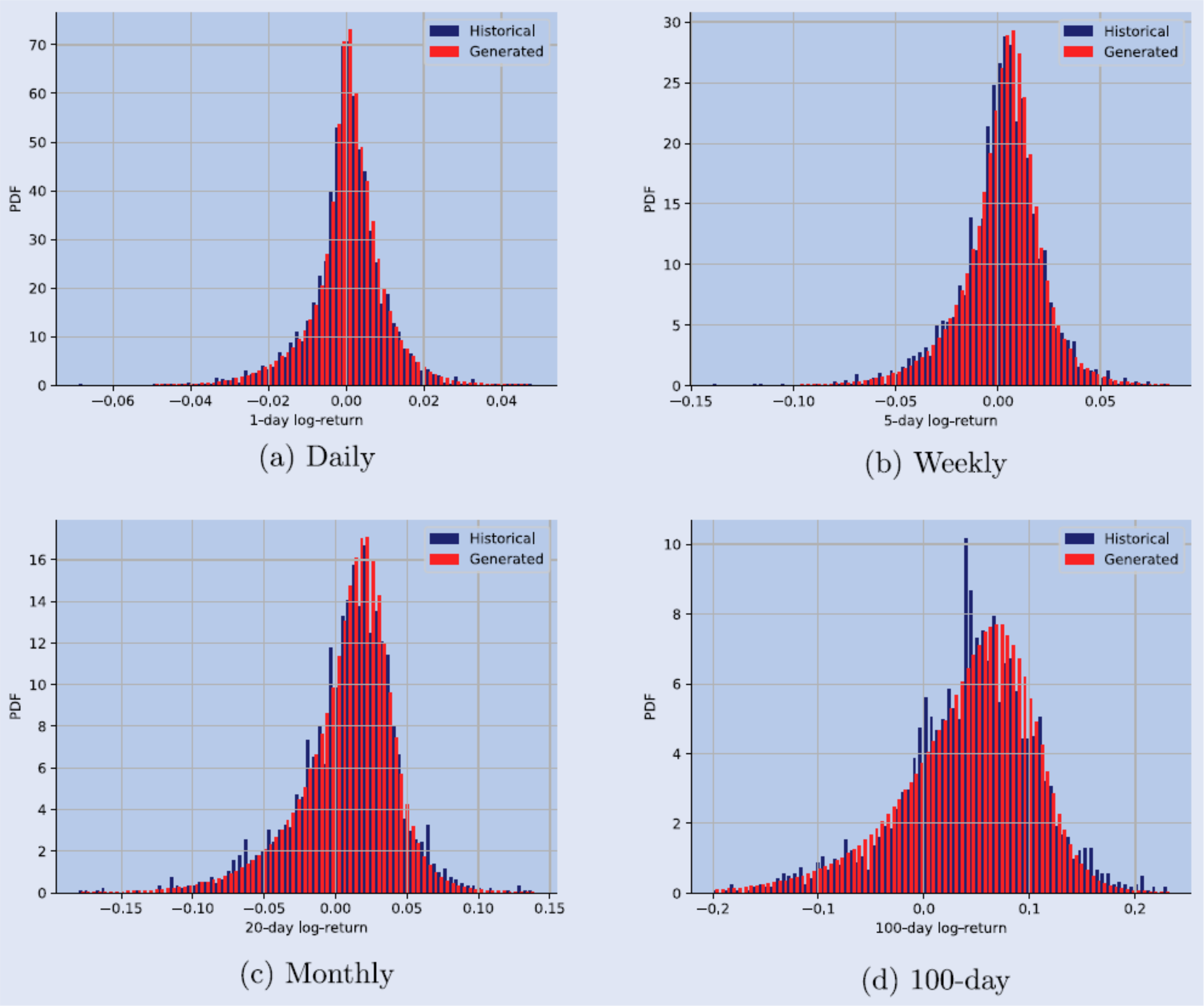


Figure A3. Comparison of generated and historical densities of the S&P500: (a) Daily, (b) Weekly, (c) Monthly and (d) 100-day.

Numerical Results

Constrained SVNN

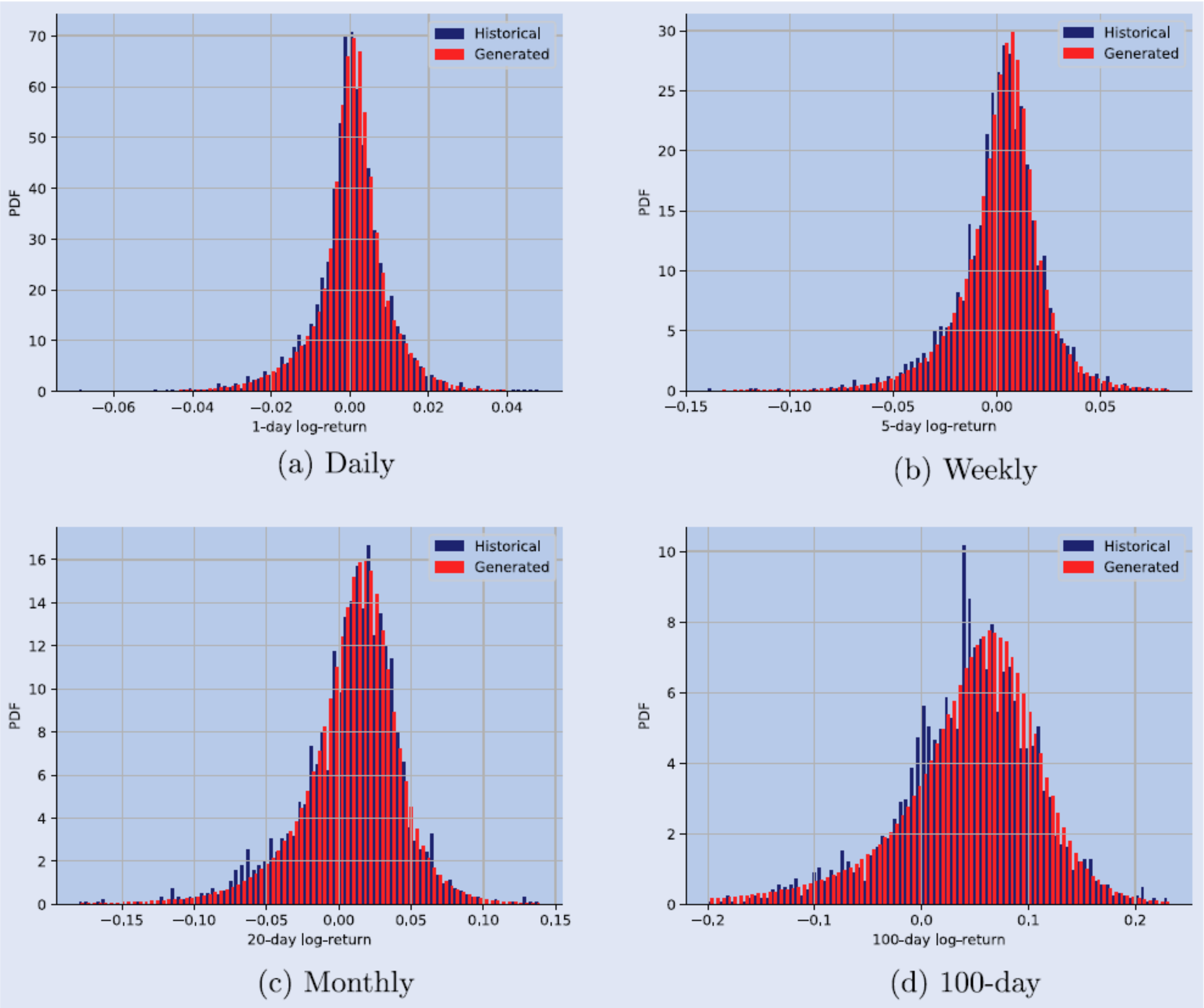


Figure A7. Comparison of generated and historical densities of the S&P500: (a) Daily, (b) Weekly, (c) Monthly and (d) 100 days.

Numerical Results

GARCH(1,1) with constant drift

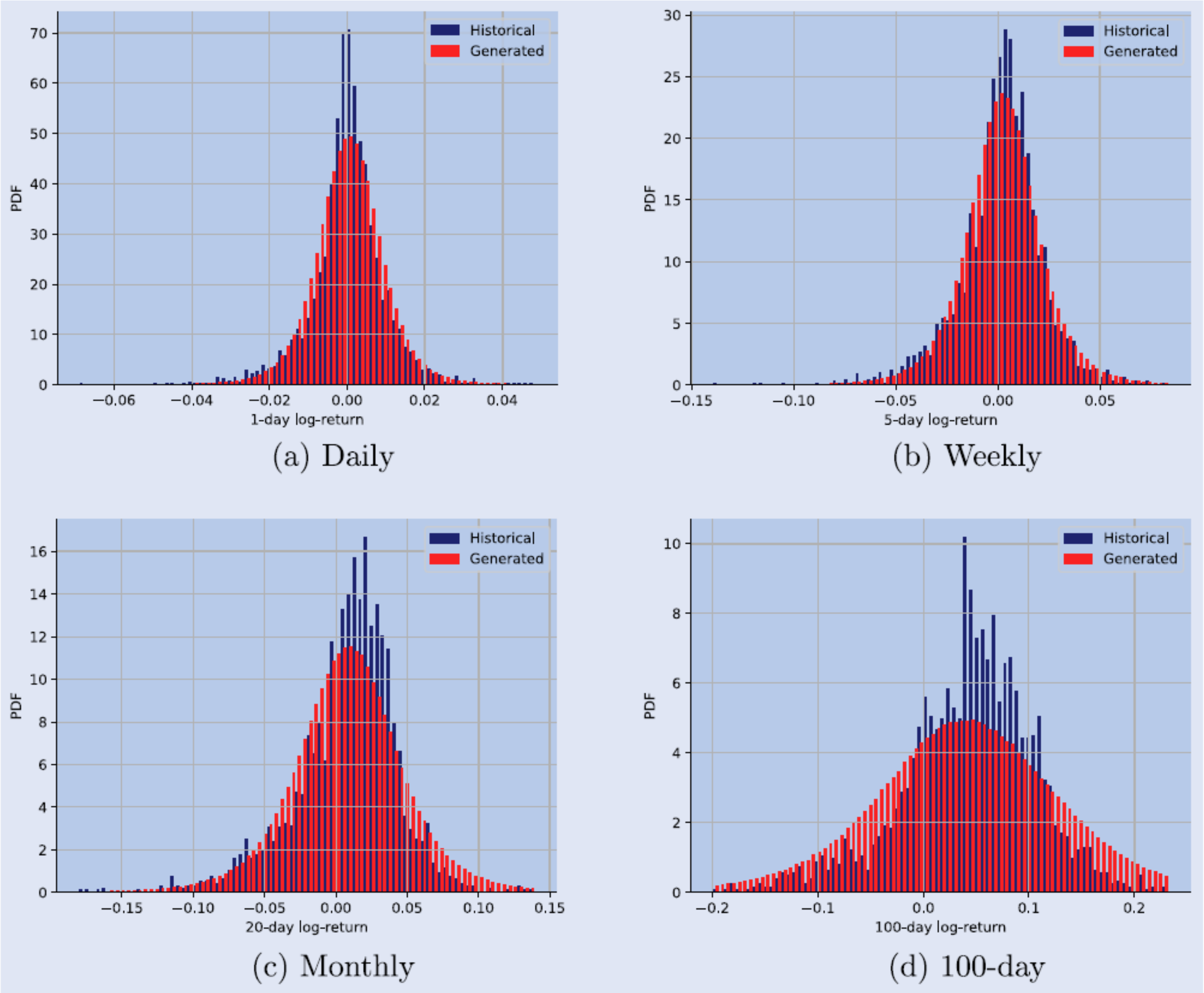
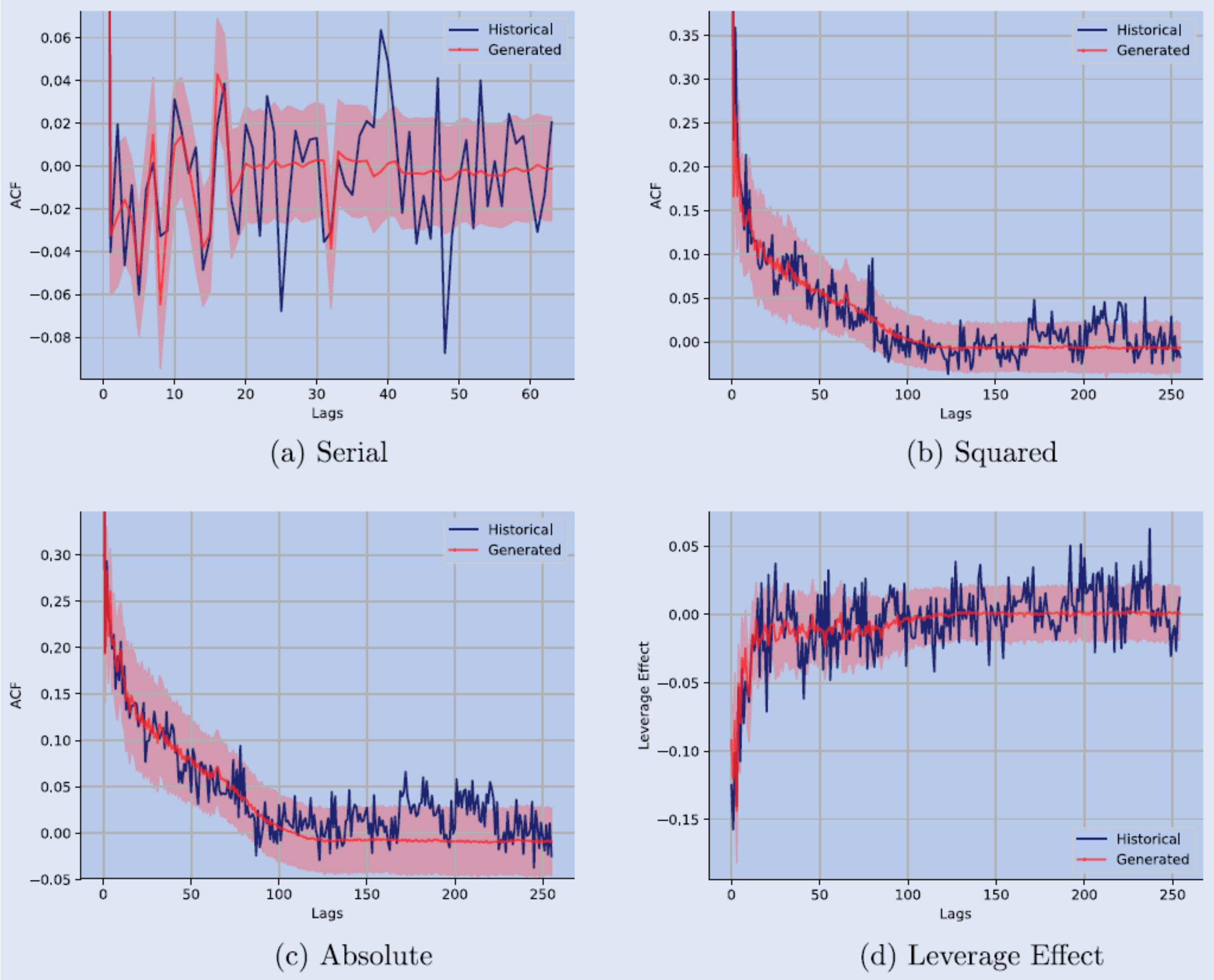


Figure A11. Comparison of generated and historical densities of the S&P500: (a) Daily, (b) Weekly, (c) Monthly and (d) 100 days.

Numerical Results

Pure TCN



The displayed ACFs corresponding to the generated returns are mean ACFs and thereby much smoother than the ACF of the real returns

Figure A4. Mean autocorrelation function of the absolute, squared and identical log returns and leverage effect: (a) Serial, (b) Squared, (c) Absolute and (d) Leverage effect.

Numerical Results

Constrained SVNN

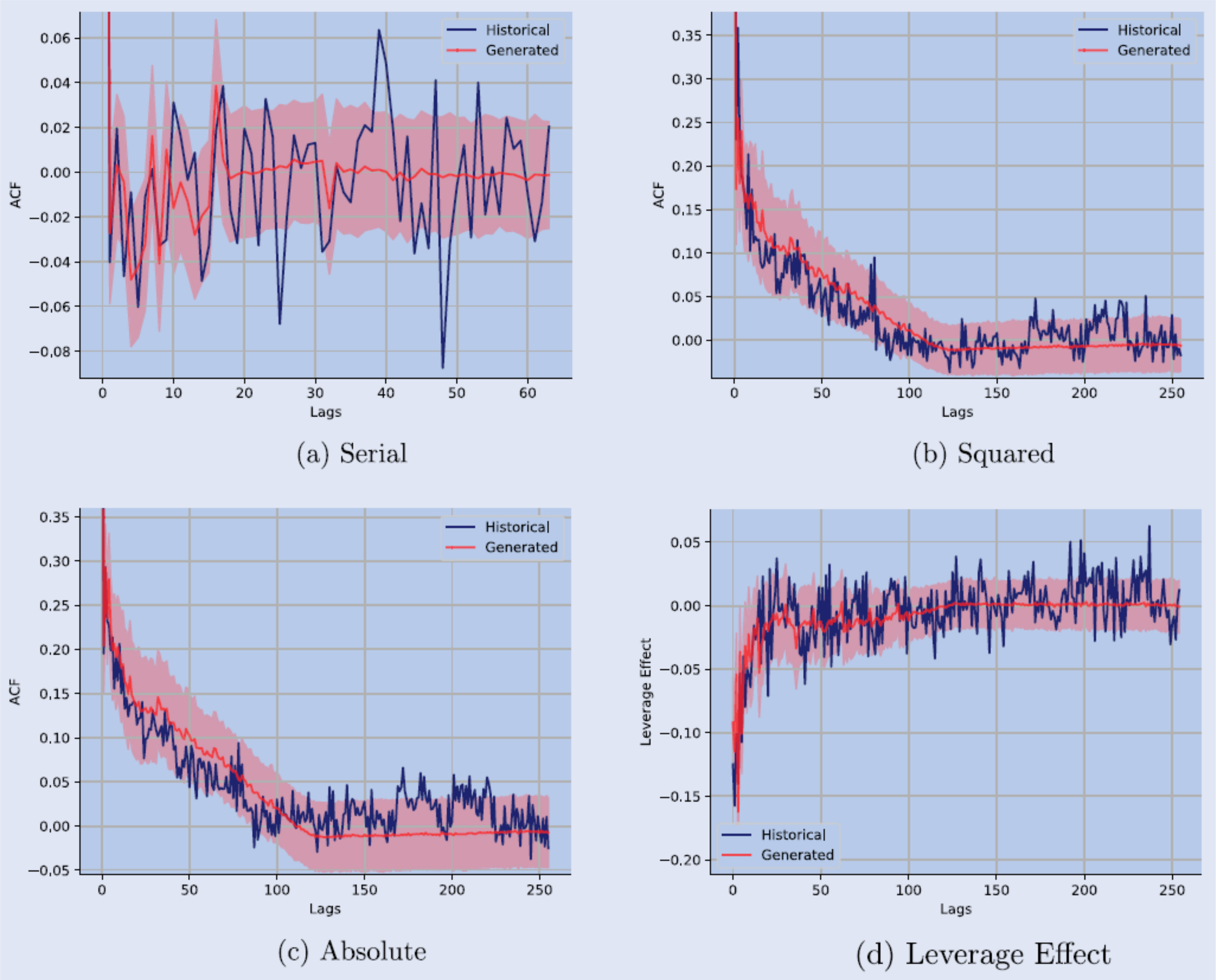
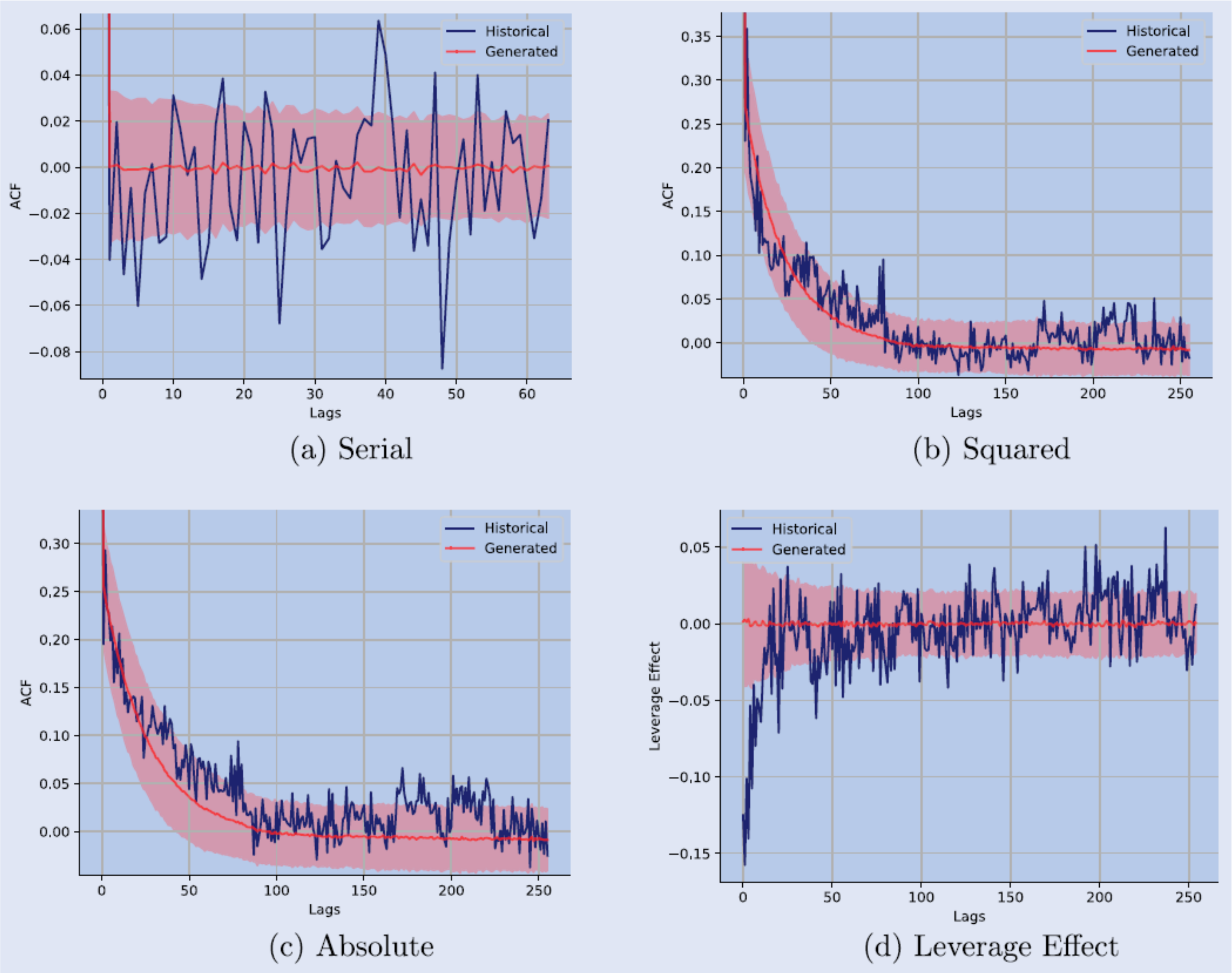


Figure A8. Mean autocorrelation function of the absolute, squared and identical log returns and leverage effect: (a) Serial, (b) Squared, (c) Absolute and (d) Leverage Effect.

Numerical Results

GARCH(1,1) with constant drift



[!t]

Figure A12. Mean autocorrelation function of the absolute, squared and identical log returns and leverage effect: (a) Serial, (b) Squared, (c) Absolute and (d) Leverage effect.

L^p-space Characterization of R_θ

Theorem 5.4 (L^p-characterization of neural networks). *Let $p \in \mathbb{N}$, $Z \in L^p(\mathbb{R}^{N_Z})$ and $g : \mathbb{R}^{N_Z} \times \Theta \to \mathbb{R}^{N_X}$ a network with parameters $\theta \in \Theta$. Then, $g_\theta(Z) \in L^p(\mathbb{R}^{N_X})$.*

Proof. Observe that for any Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$ there exists a suitable constant $L > 0$ such that

$$\|f(x) - f(0)\| \leq L \|x\| \Rightarrow \|f(x)\| \leq L \|x\| + \|f(0)\| \tag{2}$$

as $\|x\| - \|y\| \leq \|x - y\|$ for $x, y \in \mathbb{R}^n$. Now, using the Lipschitz property of neural networks (cf. Remark 3.4), we can apply Equation 2 and as Z is an element of the space $L^p(\mathbb{R}^{N_Z})$ we obtain

$$\begin{aligned}
\mathbb{E}\left[\|g_\theta(Z)\|^p\right] &\leq \mathbb{E}\left[(L \|Z\| + \|g_\theta(\mathbf{0})\|)^p\right] \\
&= \sum_{k=0}^{p} \binom{p}{k} L^k \mathbb{E}\left[\|Z\|^k\right] \|g_\theta(\mathbf{0})\|^{p-k} \\
&< \infty ,
\end{aligned}$$

where L is the networks Lipschitz constant and $\mathbf{0} \in \mathbb{R}^{N_Z}$ the zero vector. This proves the statement.

L^p-space Characterization of R_θ

Corollary 5.5. *Let R_θ be a log return NP parametrized by some $\theta \in \Theta$. Then, for all $t \in \mathbb{Z}$ and $p \in \mathbb{N}$ the random variable $R_{t,\theta}$ is an element of the space $L^p(\mathbb{R}^{N_X})$.*

Proof. The latent process Z is Gaussian i.i.d. noise. Hence, Theorem 5.4 yields $\sigma_{t,\theta}, \epsilon_{t,\theta}, \mu_{t,\theta} \in L^p(\mathbb{R}^{N_X})$. Since

$$\|R_{t,\theta}\|^p = \|\sigma_{t,\theta} \odot \epsilon_{t,\theta} + \mu_{t,\theta}\|^p \leq (\|\sigma_{t,\theta} \odot \epsilon_{t,\theta}\| + \|\mu_{t,\theta}\|)^p ,$$

we obtain using the binomial identity

$$\begin{aligned} \|R_{t,\theta}\|_p^p &= \mathbb{E}[\|R_{t,\theta}\|^p] \\ &\leq \sum_{k=0}^{p} \binom{p}{k} \mathbb{E}[\|\sigma_{t,\theta} \odot \epsilon_{t,\theta}\|^k \, \|\mu_{t,\theta}\|^{p-k}] \\ &\leq \sum_{k=0}^{p} \binom{p}{k} \left(\mathbb{E}\left[\|\sigma_{t,\theta} \odot \epsilon_{t,\theta}\|^{2k}\right] \mathbb{E}\left[\|\mu_{t,\theta}\|^{2(p-k)}\right]\right)^{\frac{1}{2}} , \end{aligned}$$

where the last inequality derives from the Cauchy-Schwarz inequality. Using the independence and the L^p-property of the volatility and innovation NP (cf. Remark 5.3), we obtain for arbitrary $q \in \mathbb{N}$ that

$$\mathbb{E}\left[\|\sigma_{t,\theta} \odot \epsilon_{t,\theta}\|^q\right] = \mathbb{E}\left[\sum_{i=1}^{N_X} |\sigma_{t,\theta,i} \, \epsilon_{t,\theta,i}|^q\right] = \sum_{i=1}^{N_X} \mathbb{E}\left[|\sigma_{t,\theta,i}|^q\right] \mathbb{E}\left[|\epsilon_{t,\theta,i}|^q\right] < \infty.$$

Thank you for listening