Machine Learning with Applications 7 (2022) 100255

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Long-term financial predictions based on Feynman–Dirac path integrals, deep Bayesian networks and temporal generative adversarial networks

Farzan Soleymani [a], Eric Paquet [b,*]

[a] Department of Mechanical Engineering, University of Ottawa, Ottawa, ON, Canada
[b] National Research Council, 1200 Montreal Road, Ottawa, ON, K1A 0R6, Canada

ARTICLE INFO

Keywords:
Temporal generative adversarial network
Time series
Financial predictions
Long short-term memory
Temporal convolutional network

ABSTRACT

This paper presents a new deep learning framework, QuantumPath, for long-term stock price prediction, which is of great significance in portfolio management and risk mitigation, especially when the market becomes volatile due to unpredictable circumstances such as a pandemic. Our approach is based on stochastic equations, the Feynman–Dirac path integral, deep Bayesian networks, and temporal generative adversarial neural networks (t-GAN). The expected financial trajectory is evaluated with a Feynman–Dirac path integral. The latter involves summing all possible financial trajectories that *could* have been taken by the financial instrument. These trajectories are generated with a t-GAN. A probability is attributed to each point of each path. The probability is a function of the Lagrangian, which is derived from a stochastic equation describing the temporal evolution of the stock. The drift and the volatility at each point, which are required in order to evaluate the Lagrangian, are predicted with a deep Bayesian neural network. Given that the evolution of a stock's price is isomorphic to a time series, our temporal GAN consists of long short-term memory (LSTM) neural networks, which introduce a memory mechanism, and temporal convolutional neural networks (TCN), which ensure causality. Stock prices are predicted over periods of twenty and thirty days for nine stocks, eight of which are included in the S&P 500 index. Our experimental results clearly demonstrate the efficiency of our approach.

The stock market is known for its erratic behaviour (Ziemba et al., 2017) with a 2020 episode resulting from the pandemic (Ashraf, 2020; Zhang et al., 2020). During the first quarter of 2020, while numerous companies experienced a sharp decline in their value, others such as Amazon, Walmart, and Tesla were sharply rising. As a result, those who invested in these companies prior to the pandemic made substantial profits while others suffered heavy losses (Nicola et al., 2020). In such contexts, stock price prediction may contribute to improving the return on investment while also mitigating the risks associated with uncertainty (Zhou et al., 2018). Generative models have the ability to learn the probability density function associated with a dataset (e.g., audio and images) (Lucic et al., 2018) while the posterior distribution may be inferred from discriminative models (Ulusoy & Bishop, 2006). Generative adversarial networks (GAN) are generative models that were first introduced by Goodfellow et al. (2014). They aim to generate synthetic data with the same probability distribution function as the training dataset. They consist of a generator that produces synthetic data and a discriminator, which evaluates the discrepancy between the synthetic data and the real data.

These networks are notably difficult to train and tend to suffer from mode collapse (He et al., 2019; Thanh-Tung & Tran, 2020). High-frequency stock market prediction based on GAN was first proposed by Zhou et al. (2018). Their network consisted of a long short-term memory (LSTM) network for the generator and a convolutional neural network (CNN) for the discriminator. Mode collapsing was reduced in Saatci and Wilson (2017) by employing Bayesian inference. Bayesian neural networks were employed both for the generator and for the discriminator, and the loss function was defined in terms of maximum likelihood. However, this approach did not guarantee convergence (Kononenko, 1989; Saatci & Wilson, 2017). Additionally, a stock market prediction framework based on GAN was proposed in Zhang et al. (2019), where the discriminator and the generator were implemented with a multi-layer perceptron (MLP) and a LSTM, respectively. The objective was to predict the daily closing prices of various stocks from the S&P 500.

In quantum mechanics, quantum state evolution has been tackled by renowned scientists such as Schrödinger, Dirac, and Feynman. Initially, Schrödinger derived a wave equation from the Hamiltonian of the system under consideration (Andreev, 2006). Later on, Dirac and Feynman proposed an alternative, but completely equivalent, formulation, based on the Lagrangian. In Dirac and Feynman's approach, time evolution is expressed in terms of a path integral which is evaluated as a weighted

* Corresponding author.
E-mail addresses: fsoleyma@uottawa.ca (F. Soleymani), Paquet@nrc-cnrc.gc.ca (E. Paquet).

https://doi.org/10.1016/j.mlwa.2022.100255
Received 20 September 2021; Received in revised form 13 December 2021; Accepted 10 January 2022
Available online 13 January 2022

sum over all possible paths or trajectories between the initial and the final states, the weight being determined by a functional of the Lagrangian associated with the system under consideration (Perepelitsa, 2018). The path integral is discrete in nature and may be evaluated with a truncated expansion (Kakushadze, 2015; Perepelitsa, 2018). Such path integrals have been employed in numerous applications, including protein folding, quantum mechanics, stochastic dynamics, quantum field theories, to mention just a few (Ingold, 2002; Linetsky, 1997).

Temporal convolutional networks (TCNs) were originally introduced by Lea et al. (2016) for video-based action segmentation. They allow a hierarchical, bi-level analysis of time series (Lea et al., 2016). Low-level features are initially extracted with a CNN, and high-level features are subsequently captured by a recurrent neural network (RNN) (Lea et al., 2016). TCNs tend to outperform standard recurrent networks when temporal series or data streams are involved (Bai et al., 2018; Chung et al., 2014; Jozefowicz et al., 2015; Pascanu et al., 2013). In addition, TCNs tend to have a longer long-term memory than RNNs, making them more suitable when remote past events are required in order to obtain accurate predictions (Bai et al., 2018; Oord et al., 2016; Wan et al., 2019).

In this paper, a new framework for financial predictions, called QuantumPath, is proposed. The expected financial trajectory for a given stock is obtained from a Feynman–Dirac path integral derived from a generic stochastic equation, this latter describing the temporal evolution of the financial instrument. The drifts and the volatilities, which are the parameters required to evaluate the probability associated with a particular realisation of a given path, are predicted by a deep Bayesian neural network. The various trajectories which result from sampling the data distribution and which are required in order to evaluate the path integral with Monte Carlo techniques are predicted using a temporal GAN. Three architectures are proposed for the t-GAN: a dense network for both the generator and the discriminator, a dense network for the generator and a LSTM network for the discriminator, and a TCN for both the generator and the discriminator.

The paper is organised as follows. Feynman–Dirac path integrals for financial predictions are introduced in Section 1. The prediction of the path integral parameters, using deep Bayesian networks, is addressed in Section 2. Financial path generations and data distribution sampling with a t-GAN are exposed in Section 3. The architecture of our t-GAN is described in Section 4. Our experimental results are reported in Section 5, while Section 6 concludes the paper.

1. Financial path prediction using stochastic equations and the Feynman–Dirac path integral

Essentially, there are two formulations of quantum mechanics: the Schrödinger (Goldin, 2008) equation, and the Feynman–Dirac equations (Truman, 1978) (The Heisenberg equation (Esteve, 2002) is an alternative formulation of the Schrödinger equation (Aizenman et al., 2006) in which the state vectors are stationary while the operators are time-dependent.) The Schrödinger and the Feynman–Dirac formulations are entirely equivalent (Wu¨thrich, 2010). The Schrödinger formulation is based on the time-dependent Schrödinger equation:

$$i\hbar \frac{\partial |S(t)\rangle}{\partial t} = \widehat{\mathcal{H}} |S(t)\rangle \tag{1}$$

in which $|S(t)\rangle$ is the time-dependent state vector and $\widehat{\mathcal{H}}$ is the Hamiltonian operator; the latter expressing the system's total energy. Whereas most quantum computing algorithms are based on the Schrödinger formulation, here we formulate the quantum algorithm in terms of Feynman–Dirac path integrals. In contrast to most approaches, the quantum formulation is not imposed *ex cathedra* but, rather, derived directly from the stochastic equations associated with the temporal evolution of the financial instruments. The temporal evolution of a security S may be characterised by a stochastic equation (Linetsky, 1997):

$$\frac{dS(t)}{dt} = \mu(t)S(t) + \sigma(t)S(t)\xi(t) \tag{2}$$

where $\mu(t)$ and $\sigma(t)$ are the drift and volatility, respectively. It is worth nothing that no hypothesis is made about the nature of these functions. The stochastic noise is defined as

$$\begin{aligned} &E[\xi(t)] = 0, \\ &E[\xi(t)\xi(t')] = \delta(t-t'), \\ &T_i \le t, t' \le T_f \end{aligned} \tag{3}$$

where $E[\cdot]$ is the mathematical expectation (Strukov & Timan, 1977) while T_i and T_f define the investment horizon. Naturally, the evolution of a security is a discrete process characterised by a 'tick,' ε, which is the smallest temporal movement admissible:

$$\begin{aligned} &\frac{S(t_{n+1}) - S(t_n)}{t_{n+1} - t_n} \simeq \alpha(t_n)S(t_n) + \sigma(t_n)S(t_n)\xi(t_n), \\ &t_{n+1} - t_n \equiv \varepsilon \quad \forall n. \end{aligned} \tag{4}$$

Instead of considering the evolution of a state vector, the Feynman–Dirac equation considers all possible financial trajectories associated with a security: a probability, determined by the Lagrangian of the financial process, is associated with every realisation. This is to be contrasted with the Hamiltonian-based Schrödinger equation; the Lagrangian and the Hamiltonian are related by a Legendre transformation (Tulczyjew, 1977). As the various realisations of a security are determined by the stochastic noise, the Feynman–Dirac equation considers all possible noise realisations. A detailed description of path integrals may be found in Masujima (2008) and Perepelitsa (2018). Therefore, the Feynman–Dirac integration metric is given by

$$\int \mathcal{D}\xi \triangleq \prod_{t=T_i}^{T_f} \int_{-\infty}^{\infty} d\xi(t). \tag{5}$$

where the integration is carried out over all possible noise values up to the investment horizon. According to the Feynman–Dirac equation, the probability of a given noise realisation is

$$\int \mathcal{D}\xi \, p[\xi] = \frac{1}{Z} \int \mathcal{D}\xi \, \exp[\mathcal{A}[\xi]] \tag{6}$$

where Z is a normalisation factor:

$$Z = \int \mathcal{D}\xi \, \exp[\mathcal{A}[\xi]] \tag{7}$$

and

$$\mathcal{A}[\xi] = \int_{T_i}^{T_f} dt \, \mathcal{L}[\xi]. \tag{8}$$

is the action associated with the stochastic noise, i.e. the time integral of the Lagrangian (Dirac, 2005). The Lagrangian may be defined as a quadratic function:

$$\begin{aligned} &\mathcal{L}[\xi] \triangleq -\frac{1}{2}\xi(t)M(t,t')\xi(t'), \\ &M(t,t') = (E[\xi(t)\xi(t')])^{-1} \Rightarrow \boxed{\mathcal{L}[\xi] = -\frac{1}{2}\xi^2(t)}. \end{aligned} \tag{9}$$

Let us define

$$z(t) \equiv \ln S(t). \tag{10}$$

Then, Eq. (2) becomes:

$$\frac{dz(t)}{dt} = -\mu(t) - \sigma(t)\xi(t). \tag{11}$$

The path integral of the Dirac delta function (Linetsky, 1997) of Eq. (11) is hence

$$\int \mathcal{D}z \, \delta\left(\frac{1}{\sigma(t)}\left(\frac{\partial z(t)}{\partial t} + \mu(t)\right) + \xi(t)\right) = 1. \tag{12}$$

As this path integral is equal to the identity, Eq. (12) may be inserted into Eq. (7) without changing its value:

$$Z = \int \mathcal{D}\xi\, p[\xi] = \int \mathcal{D}z\mathcal{D}\xi\, \delta\left(\frac{1}{\sigma(t)}\left(\frac{\partial z(t)}{\partial t} + \mu(t)\right) + \xi(t)\right) \exp[\mathcal{A}[\xi]]. \quad (13)$$

This expression may now be simplified by integrating over the stochastic noise:

$$\begin{aligned} \mathcal{D}z\, p[z] &= \frac{1}{Z'}\mathcal{D}z\, \exp[\mathcal{A}[z]], \\ \mathcal{A}[z] &= \int_{T_i}^{T_f} dt\, \mathcal{L}[z], \\ \mathcal{L}[z] &= -\frac{1}{2}\left[\frac{\frac{\partial z(t)}{\partial t} + \mu(t)}{\sigma(t)}\right]^2, \\ \int \mathcal{D}z &= \prod_{t=T_i}^{T_f} \int_{-\infty}^{\infty} dz(t), \\ Z' &= \int \mathcal{D}z\, \exp[\mathcal{A}[z]]. \end{aligned} \quad (14)$$

Such a path integral is illustrated in Fig. 1. It follows that the expectation value of a security, which is the prediction, becomes

$$E[z] \stackrel{\wedge}{=} \int \mathcal{D}z\, p[z]z = \frac{1}{Z}\int \mathcal{D}z\, \exp[\mathcal{A}[z]]z \quad (15)$$

where all the terms are defined in Eq. (14). This path integral may be evaluated with Monte Carlo techniques (Hurtado & Barbat, 1998):

$$E(z(t)) = \frac{1}{\sum_{i=1}^{N} p(z^{(i)}(t))} \sum_{i=1}^{K} z^{(i)}(t) p(z^{(i)}(t)), \quad (16)$$

$$p(z^{(i)}(t)) = \frac{1}{\sigma(t)\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\frac{\partial z^{(i)}(t)}{\partial t} - \mu(t)}{\sigma(t)}\right)^2\right] \quad (17)$$

where $z^{(i)}(t)$ is a particular realisation of the financial instrument $z(t)$. The expected value, which is also the observable, corresponds to a weighted sum over all possible financial trajectories; the weights have Gaussian probability distribution functions. In order to apply this quantum model, one should be able to predict, at each time step, the drift and the volatility as well as generating the various financial paths. Instead of utilising an *ad hoc* model, we propose to generate the financial paths with a temporal generative adversarial network, and to predict the drift and the volatility with a deep Bayesian network. Applying Monte Carlo techniques entails sampling the probability distribution associated with the data; the better the sampling, the faster the convergence, and the higher the accuracy that may be achieved. Unfortunately, these distributions are notoriously difficult to sample because of their high dimensionality (Hurtado & Barbat, 1998). By employing generative models, which learn the data distribution, we expect that both fast convergence and accuracy may be achieved with a relatively small number of paths. Experimental results shall indeed demonstrate that it is the case. Deep Bayesian neural networks are introduced in the next section.

The parameters of the path integral are predicted using deep Bayesian neural networks.

2. Deep Bayesian neural network for predicting drift and the volatility

Bayesian neural networks were introduced by David (1992). They are essentially stochastic networks in which the parameter estimation is done by Bayesian inference (Jospin et al., 2020). A Bayesian neural network may be defined as

$$p(\mathbf{y}\,|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}\,|\mathbf{x}, \omega) p(\omega\,|\,\mathbf{X}, \mathbf{Y}) d\omega \quad (18)$$

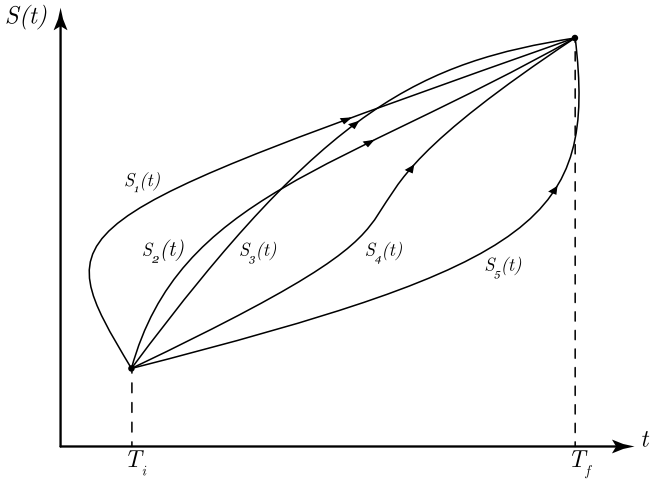

Fig. 1. Some of the paths, or trajectories, involved in evaluating a Feynman–Dirac path integral.

where $\mathbf{x}$ is the input, $\mathbf{y}$ the corresponding output, ω represents the parameters of the neural network, and $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ and $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ are the input and output training sets, respectively. Here, $p(\mathbf{y}\,|\,\mathbf{x}, \mathbf{X}, \mathbf{Y})$ is the conditional probability of the output given the input and the training set, $p(\mathbf{y}\,|\mathbf{x}, \omega)$ is the conditional probability of the output given the input and the parameters, and $p(\omega\,|\,\mathbf{X}, \mathbf{Y})$ is the posterior distribution over the training set. As opposed to a predictive standard neural network, the output of a predictive Bayesian neural network is a density probability function rather than just a single value. It thus allows for the prediction of drift and volatility. Our Bayesian neural network is a dense deep neural network, which means that

$$\omega = \{\mathbf{W}_i\}_{i=1}^{L} \quad (19)$$

where $\{\mathbf{W}_i\}_{i=1}^{L}$ are the weights associated with the various layers. The weights priors are all assumed to be Gaussian:

$$p_0(\mathbf{W}_i) = \mathcal{N}(\mathbf{W}_i; \mathbf{0}, \mathbf{I}). \quad (20)$$

The weights posterior is are not tractable, in general, but may be approximated by variational inference (Blei et al., 2017; Zhang et al., 2018). As a result, the real posterior weights posterior $p(\omega|\mathbf{X},\mathbf{Y})$ is approximated with a variational distribution $q_\theta(\omega)$. The variational parameters θ are obtained by minimising the Kullback–Leibler divergence $\mathrm{KL}[q\,\|p]$ between the variational distribution and the real posterior distribution (Amin, 2019; Chen & Shao, 1997). The Kullback–Leibler divergence is zero when both distributions are identical. Assuming that the data are statistically independent, one obtains:

$$\begin{aligned} \mathrm{KL}[q\Big\|p] &= E_q[\ln \frac{q}{p}] = \mathrm{KL}[q_\theta(\omega)\Big\| \ln p(\mathbf{Y}\,|\,\mathbf{X}, \omega)] \\ &\propto -\int_n q_\theta(\omega) \ln p(\mathbf{Y}\,|\,\mathbf{X}, \omega) d\omega + \mathrm{KL}[q_\theta(\omega)\Big\|p_0(\omega)] \\ &= -\sum_{i=1} \int q_\theta(\omega) \ln p(\mathbf{y}_i\,|\,f^\omega(\mathbf{x}_i)) d\omega + \mathrm{KL}[q_\theta(\omega)\Big\|p_0(\omega)] \end{aligned} \quad (21)$$

where $f^\omega(\cdot)$ is a dense deep neural network with

$$\theta = \{\mathbf{M}_i\}_{i=1}^{L} \quad (22)$$

being the corresponding weights. Deterministic neural networks with stochastic regularisation, such as dropouts (Ba & Frey, 2013), may be interpreted as Bayesian neural network approximations (Kononenko, 1989; Springenberg et al., 2016). Indeed, applying a stochastic reg-

ularisation technique is equivalent to multiplying the neural network weights by random noise:

$$\mathbf{W}_i = \varepsilon_i \mathbf{M}_i. \tag{23}$$

The resulting stochastic matrix samples the posterior distribution over the Bayesian neural network weights (Kononenko, 1989; Mullachery et al., 2018). The KL[$q \parallel p$] divergence may be generalised by means of Amari's α-divergence (Cai et al., 2020; Hernandez-Lobato et al., 2016), this latter being defined as

$$\mathrm{D}_\alpha[p \parallel q] = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(\omega)^\alpha q(\omega)^{1-\alpha} d\omega\right). \tag{24}$$

Amari's α-divergence encompasses numerous metrics which include the KL[$q \parallel p$] divergence ($\alpha = 0$), and the Hellinger distance ($\alpha = 1/2$), amongst others. Therefore, the Kullback–Leibler divergence in Eq. (21) may be replaced with Amari's α-divergence. The resulting energy function (a cost or loss function) is called a black-box α energy function (Hernandez-Lobato et al., 2016). Such an energy function provides a better approximation for the marginal likelihood, resulting in a model that is less biased and which better fits the data distribution (Habeck, 2012). To make the text more readable, the following notation shall be adopted henceforth:

$$p(\omega) = \frac{1}{Z} p_0(\omega) \prod_n f_n(\omega), \tag{25a}$$

$$p(\omega) \equiv p(\omega \mid \mathbf{X}, \mathbf{Y}) \tag{25b}$$

$$f_n(\omega) \equiv p(\mathbf{y}_n \mid \mathbf{x}_n, \omega), \tag{25c}$$

$$Z = p(\mathbf{Y} \mid \mathbf{X}). \tag{25d}$$

In the most general case, the black-box α energy function is given by

$$KL[q \Big\| p] \to \mathrm{D}_\alpha[q \Big\| p] \Rightarrow \boxed{\mathcal{L}_\alpha(q) = -\frac{1}{\alpha} \sum_n \ln E_q\left[\left(\frac{f_n(\omega) p_0(\omega)^{1/N}}{q(\omega)^{1/N}}\right)^\alpha\right]}. \tag{26}$$

This equation reduces to Eq. (21) when $\alpha \to 0$:

$$\lim_{\alpha \to 0} \mathcal{L}_\alpha(q) \equiv \mathcal{L}_0(q) = \mathrm{KL}[q \Big\| p_0] - \sum_n E_q[\ln f_n(\omega)]. \tag{27}$$

Yet, Eq. (26) is computationally prohibitive ($\mathcal{O}(N)$). In order to solve this problem, Santana and Hernández-Lobato (2020) proposed a reparametrisation of the black-box α energy function in order to reduce the computational burden. Firstly, the posterior distribution is approximated with a free-form cavity distribution:

$$q(\omega) = \frac{1}{Z_q} \tilde{q}(\omega) \left(\frac{\tilde{q}(\omega)}{p_0(\omega)}\right)^{\frac{\alpha}{N-\alpha}}. \tag{28}$$

When this approximation is substituted in Eq. (26), the energy function reduces to

$$\mathcal{L}_\alpha(q) = -\frac{1}{\alpha} \sum_n \ln \int \left(\frac{1}{Z_q} \tilde{q}(\omega) \left(\frac{\tilde{q}(\omega)}{p_0(\omega)}\right)^{\frac{\alpha}{N-\alpha}}\right)^{1-\frac{\alpha}{N}} \times p_0(\omega)^{\alpha/N} f_n(\omega)^\alpha d\omega. \tag{29}$$

As $\alpha/\mathrm{N} \simeq 0$ in general, this expression may be further simplified:

$$\lim_{\frac{\alpha}{N} \to 0} \mathcal{L}_\alpha(q) \approx \tilde{\mathcal{L}}_\alpha(q) = \mathrm{KL}[q \Big\| p_0] - \frac{1}{\alpha} \sum_n E_q[\ln f_n(\omega)^\alpha]. \tag{30}$$

Since our Bayesian network is employed in a regression/prediction context, we have

$$f_n(\omega) \propto p(\mathbf{y}_n \mid \mathbf{x}_n, \omega) \propto \exp[-\ell(\mathbf{y}_n, f^\omega(\mathbf{x}_n))] \tag{31}$$

where $\ell(\mathbf{y}_n, f^\omega(\mathbf{x}_n))$ is a loss function which measures the discrepancy between the prediction and the real data. The energy function may be further simplified by assuming a quadratic loss function:

$$\ell(\mathbf{y}_n, f^\omega(\mathbf{x}_n)) = \frac{\tau}{2} \left\| \mathbf{y}_n, f^\omega(\mathbf{x}_n) \right\|_2^2 \tag{32}$$

and a Gaussian likelihood

$$\mathbf{y} \sim \mathcal{N}(\mathbf{y}_n; f^\omega(\mathbf{x}_n), \tau^{-1}\mathbf{I}). \tag{33}$$

If this function is sampled with a Monte Carlo technique, one finally obtains:

$$\mathcal{L}_\alpha^{\mathrm{MC}}(q) = -\frac{1}{\alpha} \sum_n \ln \frac{1}{K} \sum_k \exp[-\frac{\alpha\tau}{2} \left\| \mathbf{y}_n, f^{\tilde{\omega}_k}(\mathbf{x}_n) \right\|_2^2] + \frac{ND}{2} \log \tau + \sum_i p_i \left\| \mathbf{M}_i \right\|_2^2, \tag{34}$$

$$\tilde{\omega}_k \sim q(\omega)$$

where the $\{\tilde{\omega}_k\}$ are sampled from the variational distribution $q(\omega)$, and the $\{\mathbf{M}_i\}$ are the dropout matrices as defined in Eq. (23). The financial trajectories are also needed to evaluate the path integral; these are generated using a t-GAN.

3. Generating financial trajectories with a temporal GAN

In order to evaluate the path integral, multiple financial trajectories must be generated. Instead of assuming an ad hoc model, we propose to learn this model, and to generate these trajectories with a temporal generative adversarial network (Goodfellow et al., 2014; Zhao et al., 2016). Indeed, as the evaluation of the path integral with a Monte Carlo technique entails sampling the underlying data distribution, we propose to learn and sample this distribution with a GAN. As the generated trajectories will be more realistic, the Monte Carlo calculation is more likely to converge rapidly with high accuracy. In its original form, generative neural networks consists of a generator and a discriminator. The generator creates financial trajectories from a uniform random function, while the discriminator ensures that these trajectories are as realistic as possible. In order to meet these requirements, the energy function was originally defined (Goodfellow et al., 2014) as

$$\mathcal{L}_G = \min_{\theta_G} \max_{\theta_D} E_{\mathbf{x} \sim p_D}[ln D_{\theta_D}(\mathbf{x})] + E_{\mathbf{z} \sim p_G}[ln(1 - D_{\theta_D}(G_{\theta_G}(\mathbf{z})))], \quad \mathbf{z} \sim \mathcal{U} \tag{35}$$

where the discriminator $D_{\theta D}(\cdot)$ is a deep neural network with parameters θ_D, $G_{\theta G}(\cdot)$ is the generator, a deep neural network with parameters θ_G, and $\mathbf{z}$ is a random vector sampled from a uniform distribution $\mathcal{U}$. The two networks may be assimilated to players in a minimax game (Du & Pardalos, 2013; Wang et al., 2017). These networks are notoriously difficult to train, and tend to be unstable (Thanh-Tung & Tran, 2020). Therefore, the Wasserstein generative adversarial network (Arjovsky et al., 2017) has been proposed as an alternative. During the learning process, this network minimises an approximation of the earth mover's distance (Weng, 2019) based on the Kantorovich–Rubinstein duality (Adler & Lunz, 2018; Deshpande et al., 2018). These networks tend to be easier to train, and more stable, while avoiding mode dropping (Bau et al., 2019). The discriminator of the Wasserstein generative adversarial network must satisfy a K-Lipschitz constraint (Arjovsky et al., 2017; Liu, 2018):

$$\left| D(\mathbf{x}_i) - D(\mathbf{x}_j) \right| \leq K \left| \mathbf{x}_i - \mathbf{x}_j \right|, \forall \mathbf{x}_i, \mathbf{x}_j \tag{36}$$

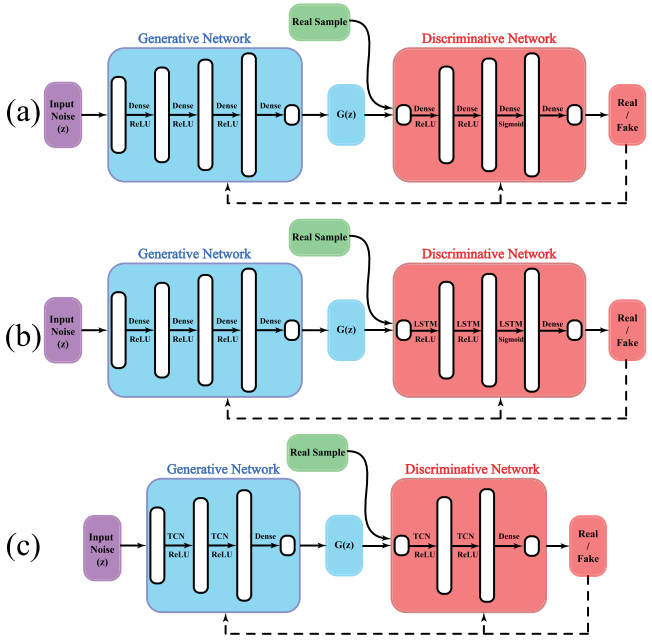

Fig. 2. GAN architectures: (a) Dense network–dense network, (b) Dense network–LSTM, (c) TCN–TCN.

where K is a constant. The Wasserstein loss (cost or objective) function (Dukler et al., 2019; Frogner et al., 2015), which approximates the earth mover's distance, has the form:

$$\mathcal{L}_W = \min_{\theta_G} \max_{\theta_D} E_{\mathbf{x}\sim p_D}[D(\mathbf{x})] + E_{\mathbf{z}\sim p_G}[D(G(\mathbf{z}))], \quad \mathbf{z} \sim \mathcal{U} \tag{37}$$

If K is set to one, the K-Lipschitz condition may be enforced by clipping the weight matrices' singular values to one (Saito et al., 2017):

$$\mathbf{W}_i = \mathbf{U}_i \Sigma_i \mathbf{V}_i^{\dagger}, \quad \Sigma = \mathrm{diag}(\sigma_1, \dots \sigma_n), \quad \sigma_j > 1 \Rightarrow \sigma_j = 1 \tag{38}$$

which implies that the spectral norms of the weights matrices are always smaller than, or equal to, one.

Financial trajectories are a particular form of time series. In their original form, generative adversarial networks were designed to handle static data such as images, not for temporally varying entities such as time series. For this reason, we propose to employ a temporal generative adversarial network for generating the financial paths (Kakushadze, 2015; Linetsky, 1997). Let us assume that the price of the securities are generated over the discrete-time interval $[t_1, t_2, \dots t_T]$, which results in T realisations of the security being considered. The t-GAN consists of two generators. The first generator, named the temporal generator, takes a latent variable $\mathbf{z}_0$ as an argument, and generates a latent-variables time series $[\mathbf{z}_1^1, \dots, \mathbf{z}_1^T]$, in which each temporal latent variable corresponds to a particular realisation of the financial trajectory. The latent variable $\mathbf{z}_0$ is drawn from a uniform distribution U. The second generator, named the financial generator, generates a particular instance of the financial trajectory from the latent variable $\mathbf{z}_0$ and from a particular instance of the temporal latent variable: $[G_1(\mathbf{z}_0, \mathbf{z}_1^1), \dots, G_1(\mathbf{z}_0, \mathbf{z}_1^T)]$.

As a result, the Wasserstein loss function associated with the t-GAN becomes:

$$\begin{aligned}\mathcal{L}_T = \min_{\theta_{G_0}, \theta_{G_1}} \max_{\theta_D} & E_{[\mathbf{x}^1,\dots,\mathbf{x}^T]\sim p_D}[D([\mathbf{x}^1, \dots, \mathbf{x}^T])] \\ & + E_{\mathbf{z}_0\sim p_{G_0}}[D([G_1(\mathbf{z}_0, \mathbf{z}_1^1), \dots, G_1(\mathbf{z}_0, \mathbf{z}_1^T)])], \\ & \mathbf{z}_0 \sim \mathcal{U}, \\ & G(\mathbf{z}_0) = [\mathbf{z}_1^1, \dots, \mathbf{z}_1^T]\end{aligned} \tag{39}$$

It should be noted that, while $\mathbf{z}_1$ varies in time, $\mathbf{z}_0$ is stationary in order to ensure consistency in between generated instances. As mentioned earlier, the K-Lipschitz condition is enforced with singular value clipping (Saito et al., 2017). In the next section, three architectures are proposed for the t-GAN.

4. GAN architectures

Three architectures are proposed for the t-GAN, illustrated in Fig. 2. The first architecture consists of two dense networks: one for the generator and one for the discriminator. The second architecture involves a dense network and a LSTM. The last architecture combines two temporal convolutional networks (TCN).

As for the Dense–Dense GAN, both the generator and the discriminator consist of four fully connected layers having 5, 10, 20, and 1 neurons respectively. The generator creates financial trajectories from a uniform random function, while the discriminator ensures that these trajectories are as realistic as possible. The discriminator is trained with a binary cross-entropy loss function, which aims to distinguish between fake and real samples.

Algorithm 1 Dense-Dense GAN

Input: financial time series, uniform distribution.
Output: Predicted trajectory for financial instrument
Initialization:
while i ≤ n **do**

- **Generative Network**
  1. **Dense layer** {units = 5, act-fcn = ReLU}
  2. **Dense layer** {units = 10, act-fcn = ReLU}
  3. **Dense layer** {units = 20, act-fcn = ReLU}
  4. **Dense layer** {units = 1 }
- **Discriminative Network**
  1. **Dense layer** {units = 5, act-fcn = ReLU}
  2. **Dense layer** {units = 10, act-fcn = ReLU}
  3. **Dense layer** {units = 20, act-fcn = ReLU}
  4. **Dense layer** {units = 1, act-fcn = Sigmoid}
  5. **Model** {loss = binary crossentropy, optimizer = Adam}

In the case of the Dense-LSTM architecture, the GAN consists of a generator (comprising four fully connected layers consisting of 10, 20, 40 and 1 neurons respectively) and a discriminator (comprising three LSTM layers having 10, 40 and 80 neurons and of an output neuron). As for the Dense-LSTM architecture, the discriminator distinguishes between fake and real samples. The LSTM layers are capable of capturing sequential patterns in financial time series, thus forcing the generator to create realistic fake samples.

Algorithm 2 Dense-LSTM GAN

Input: financial time series, uniform distribution.
Output: Predicted trajectory for financial instrument
Initialization:
while $i \leq n$ do

- **Generative Network**
  1. **Dense layer** {units = 10, act-fcn = ReLU}
  2. **Dense layer** {units = 20, act-fcn = ReLU}
  3. **Dense layer** {units = 40, act-fcn = ReLU}
  4. **Dense layer** {units = 1 }
- **Discriminative Network**
  1. **Dense layer** {units = 10, act-fcn = ReLU}
  2. **Dense layer** {units = 40, act-fcn = ReLU}
  3. **Dense layer** {units = 80, act-fcn = ReLU}
  4. **Dense layer** {units = 1, act-fcn = Sigmoid}
  5. **Model** {loss = binary crossentropy, optimizer = Adam}

In this architecture, the generator has two TCN layers, consisting of 10 neurons each, and an output neuron. In contrast, the discriminator's two TCN layers consist of 6 neurons each, followed by an output neuron. Both networks employ causal padding to prevent information leakage. As for LSTM, TCNs learn sequential patterns and long-term behaviours. When compared to LSTMs, TCNs tend to have a greater long-term memory. In our study, they outperform the other networks for long-term predictions (20 and 30 days).

Algorithm 3 TCN-TCN GAN

Input: financial time series, uniform distribution.
Output: Predicted trajectory for financial instrument
Initialization:
while $i \leq n$ **do**

- **Generative Network**
  1. **TCN layer** {units = 10, act-fcn = ReLU, padding = causal}
  2. **TCN layer** {units = 10, act-fcn = ReLU, padding = causal}
  3. **Dense layer** {units = 1}
- **Discriminative Network**
  1. **TCN layer** {units = 6, act-fcn = ReLU, padding = causal}
  2. **TCN layer** {units = 6, act-fcn = ReLU, padding = causal}
  3. **Dense layer** {units = 1, act-fcn = Sigmoid}
  4. **Model** {loss = binary crossentropy, optimizer = Adam}

In a time series, such as a financial trajectory, time flows from the past to the future. This causal directionality, which is of fundamental importance, is inherently absent from a standard convolutional neural network, so that the future may influence the past! This problem may be circumvented by employing a TCN (Wan et al., 2019). Our TCN, which is inspired by Bai et al. (2018), consists of two dilated causal convolution layers as described in Fig. 3. Combining residual and dilated convolution layers provides the ability to make predictions from both recent and past events (long-term memory) while enforcing causality (Bai et al., 2018). The dilated convolution operator $F(s)$ over a time series s is defined as (Bai et al., 2018)

$$F(s) = (\mathbf{x}_{*_d} f)(s) = \sum_{i=0}^{k-1} f(i).\mathbf{x}_{s-i.d} \tag{40}$$

where d is the dilation factor, k is the filter size, and s - $i.d$ explains the direction of the past. The dilation factor determines the scale of the convolution stride. For instance, $d = 1$ corresponds to a standard neural network. In order to avoid a vanishing or an explosion of the gradient, the TCN employs a residual module (Bai et al., 2018), this latter being illustrated in Fig. 3(b). Essentially, the residual module consists of a dilated causal convolution, followed by weight normalisation and a non-linear activation function (ReLU).

The whole procedure for price predictions is summarised in Algorithm 4. Stocks are first selected while their prices, over a given time period, are retrieved. For each stock, the time series is partitioned into a training set and a test set; the test set consists of either the last twenty or last thirty days of the time series. A set of 1024 randomly generated Bayesian networks, as described in Section 2, are trained with Eq. (34). The best network is selected to predict the drift and volatility associated with the prices. From the drifts and the volatilities, the Lagrangian is formed according to Eq. (14), the probability associated with each point of each financial path being a function of the Lagrangian. The predictions, which correspond to the expectation values, are obtained from Eq. (15). The path integral is evaluated according to Monte Carlo techniques as defined in Eq. (16): as the various paths are generated with a t-GAN, the probability distribution associated with the stocks is sampled directly by the GAN.

5. Experimental results

Our dataset consisted of nine stocks, eight of which are included in the S&P 500 index, namely Apple (AAPL), Advanced Micro Devices (AMD), Amazon (AMZN), Alphabet, better known as Google (GOOG), Microsoft (MSFT), NVidia (NVDA), Pfizer (PFE), Shopify (SHOP), and Walmart (WMT). These stocks are drawn from various sectors, such as high-technology, retail, e-commerce, and pharmaceutical, among others. Therefore, the proposed approach may be evaluated against diverse financial ecosystems. The only parameters are the start and end dates which were set to 1 January 2019 and 31 December 2020 respectively. Therefore, the time series contain data from both before and during the Covid-19 pandemic. For each tick, the data consist of open, close, high and low prices. The low, high and closing prices are normalised with respect to the opening price (Soleymani & Paquet, 2020). Our aim is to predict stock prices over periods of twenty days and thirty days. For the twenty-day prediction period, the training set consisted of the first 479 trading days, while the last twenty days' records were assigned to the test set. For the thirty-day prediction period, the training dataset consisted of the first 469 trading days, and the last thirty days' records constituted the test set. The chosen trading period was during the Covid-19 crisis, which makes the prediction task more challenging due to the economical and financial instability induced by the pandemic. The financial time series were retrieved directly with Mathematica with the FinancialData function.

The calculations were performed as follows: a total of 1024 dense Bayesian neural networks, as described in Section 2, were generated randomly. Each of these networks consisted of 1 to 5 layers with 10 to 200 neurons per layer, 3 choices for the activation function (ramp, SELU, and tanh), and either a homoscedastic (all the random variables have the same variance) or heteroscedastic (the random variables do

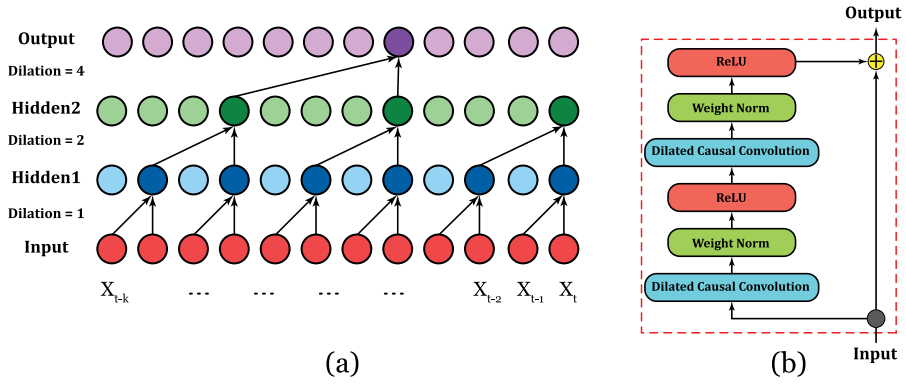

Fig. 3. (a) Illustration of a temporal convolution with dilation factors $d = 1, 2, 4$ and a filter size of $k = 2$ (b) Architecture of the residual block associated with the temporal convolutional network.

Algorithm 4 QuantumPath

Input: Prediction period T
Output: predicted financial trajectory
Data: m : number of neurons per layer (BNN): 10–200
Data: l :number of hidden layers (BNN): 1–5
Data: n : number of randomly generated Bayesian network architectures: 1024
Data: err_mod: error model (BNN): homoscedastic or heteroscedastic
Data: act_fcn: activation function (BNN): ramp, SELU and Tanh

1. Prediction of drifts and volatilities with a deep Bayesian network:
   (a) Generate randomly n BNN architectures by varying the number of :
      - Neurons per layer (1-m);
      - The number of hidden layers (1-l);
      - The activation functions (act_fcn);
      - The error model (err_mod)
   (b) Train each generated network with Adam, with the loss function described by Eq. 34.
   (c) Select the most accurate network.
   (d) Employ the network to predict the drifts and volatilities.
2. Prediction of the financial trajectory:
   (a) Train the t-GAN with Adam, employing the loss function described by Eq. 39.
   (b) Generate financial trajectories with the t-GAN (sampling of the data distribution).
   (c) Evaluate the expected predicted financial trajectory, with a Feynman–Dirac path integral (Eq. 14 –16), with Monte Carlo techniques (Eq. 16 –17), from the financial trajectories generated by the t-GAN, and the drifts and volatilities predicted by the BNN.

not share the same variance) error model. The models were trained in parallel on an HP Apollo System with 40 Dual Intel Xeon Gold 6149 processors, one Nvidia Tesla V100 (32 GB) and 968 GB of memory; one model for each stock. For each particular stock, the best model was selected for predicting the drifts and volatilities over the test periods, that is, the last twenty and thirty days, respectively.

The deep Bayesian network was implemented with Mathematica while the GANs were implemented with TensorFlow 2.2 and Python 3.7. The code for the GANs was executed on a HP Z6 workstation with two Intel Xeon Silver 4114 CPUs, 176 GB of RAM and a NVIDIA GeForce RTX 3090 GPU with 24 GB of RAM. The total execution time was 0.2 h including both training and predictions.

The drifts and the volatilities were substituted into the Lagrangian in order to evaluate the occurrence probability for each day of each financial trajectory with the help of Eq. (14). Afterwards, all three GANs were trained with the training set using the Adam algorithm (Kingma & Ba, 2015). Once trained, they were employed to generate one thousand financial trajectories or paths for the evaluation of the path integral. The mathematical expectations of the stock prices over the test period, which are inherently the predictions made by our QuantumPath framework, were evaluated with the help of Eqs. (14) and (15). The relative errors for the predictions over a period of twenty and thirty days, for all three GAN architectures, are reported in Figs. 4 and 5 respectively. These results demonstrate the efficiency of our QuantumPath framework.

The maximum relative errors, for all three GAN architectures, are reported in Fig. 6 for prediction periods of both 20 and 30 days. From Figs. 4–6, it may be concluded that the TCN–TCN GAN slightly outperforms the others, while the dense network–LSTM GAN presents the lowest performances. These results point to the importance of the long-term memory for price predictions.

The proposed method has been evaluated in low and high volatility conditions. Indeed, because of the start and end dates, 1 January 2019 and 31 December 2020 respectively, the networks were trained both with pre-pandemic and pandemic time series while the predictions were made during a volatile and uncertain period strongly affected by exogenous factors such as Covid-19 (Cervelló-Royo & Guijarro, 2020; He et al., 2020; Nti et al., 2020; Soleymani & Paquet, 2021; Zhang et al., 2020). In order to determine the impact of the time horizon on the accuracy, the RMSV was evaluated for each stock over periods of 5, 10, 15, and 20 days. The results are reported in Tables 1–3. The tables show that the RMSV remains relatively stable, or at least increases only slowly, as the investment horizon increases.

In order to compare our approach to classical ones, the financial predictions have also been performed with an autoregressive integrated

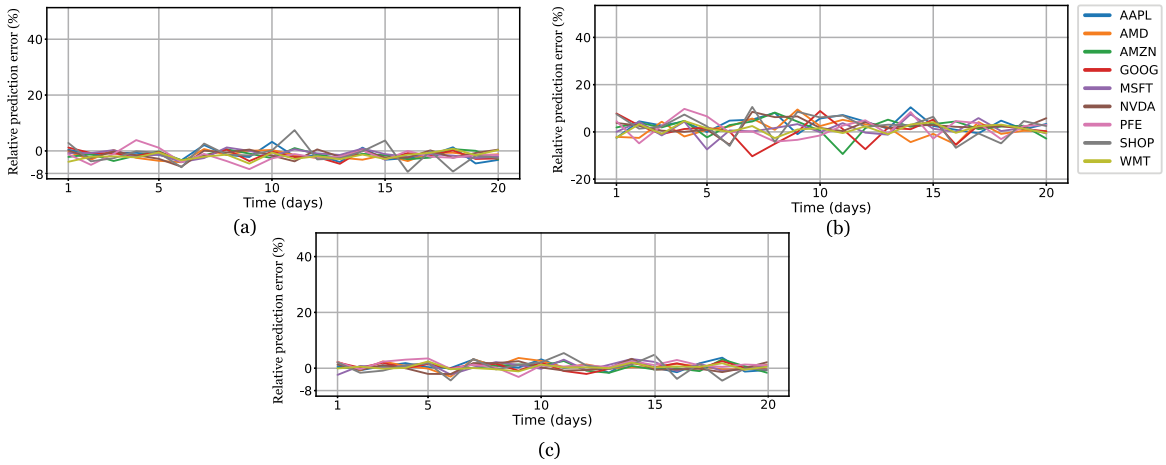

Fig. 4. Relative errors for price prediction for all nine stocks over a period of 20 days: (a) Dense network–dense network GAN, (b) Dense network–LSTM GAN, (c) TCN–TCN GAN.

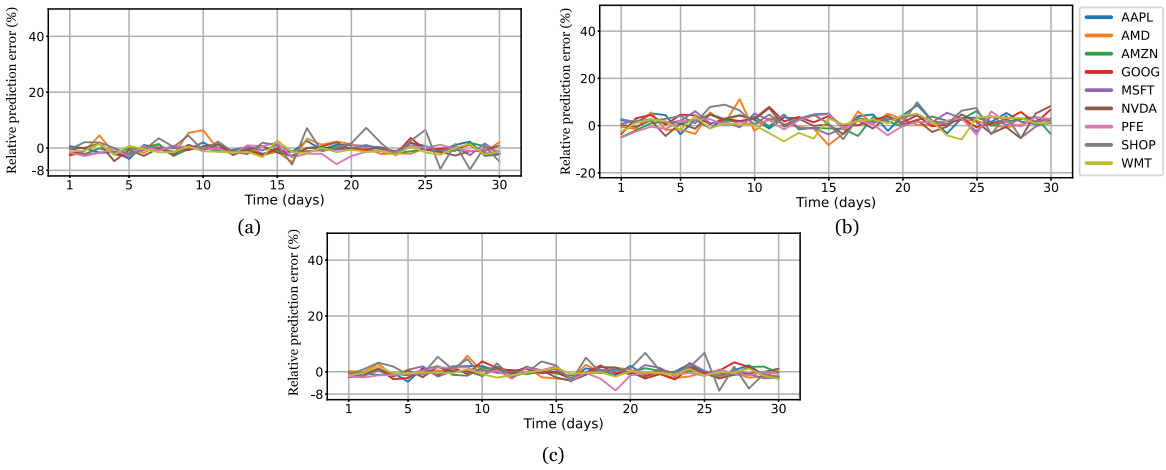

Fig. 5. Relative errors for price prediction for all nine stocks over a period of 30 days: (a) Dense network–dense network GAN, (b) Dense network–LSTM GAN, (c) TCN–TCN GAN.

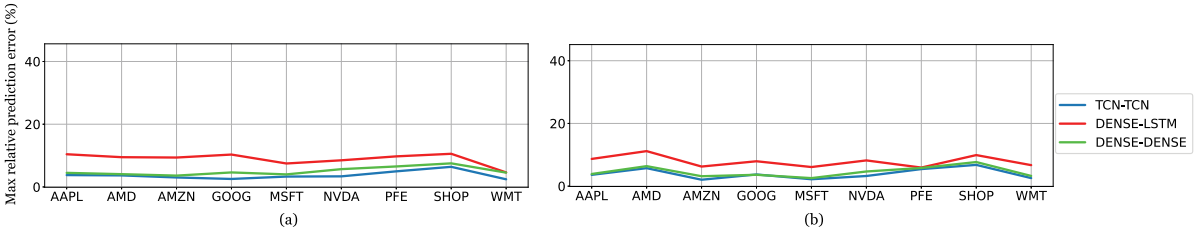

Fig. 6. Maximum relative error for price predictions for the three GAN architectures: (a) 20day period, (b) 30-day period.

Table 1
RMSV on 5, 10, 15, 20 and 30-day predictions for the Dense–Dense GAN architecture.

Stocks/Days	5	10	15	20	30
AAPL	0.025	0.035	0.035	0.040	0.308
AMD	0.026	0.021	0.024	0.032	0.269
AMZN	0.023	0.029	0.025	0.030	0.310
GOOG	0.017	0.017	0.017	0.029	0.314
MSFT	0.012	0.019	0.025	0.032	0.344
NVDA	0.011	0.035	0.041	0.035	0.364
PFE	0.011	0.022	0.037	0.042	0.340
SHOP	0.043	0.074	0.079	0.081	0.385
WMT	0.009	0.019	0.021	0.048	0.291

Table 2
RMSV on 5, 10, 15, 20 and 30-day predictions for the Dense-LSTM GAN architecture.

Stocks/Days	5	10	15	20	30
AAPL	0.094	0.060	0.063	0.148	0.081
AMD	0.076	0.075	0.081	0.080	0.139
AMZN	0.094	0.045	0.133	0.072	0.091
GOOG	0.083	0.069	0.088	0.076	0.128
MSFT	0.150	0.055	0.078	0.079	0.077
NVDA	0.075	0.057	0.097	0.096	0.096
PFE	0.140	0.059	0.132	0.062	0.091
SHOP	0.125	0.092	0.102	0.151	0.100
WMT	0.065	0.060	0.118	0.088	0.114

Table 3
RMSV on 5, 10, 15, 20 and 30-day predictions for the TCN–TCN GAN architecture.

Stocks/Days	5	10	15	20	30
AAPL	0.022	0.032	0.028	0.031	0.047
AMD	0.020	0.027	0.030	0.052	0.054
AMZN	0.021	0.025	0.026	0.025	0.041
GOOG	0.015	0.024	0.020	0.027	0.030
MSFT	0.011	0.018	0.026	0.024	0.033
NVDA	0.016	0.042	0.023	0.038	0.044
PFE	0.016	0.018	0.033	0.055	0.051
SHOP	0.038	0.075	0.068	0.070	0.096
WMT	0.009	0.028	0.017	0.021	0.045

Table 4
RMSV on 30-day predictions for GARCH, ARIMA, OU, and QuantumPath.

Stocks/Algorithm	QuantumPath	GARCH	ARIMA	OU
AAPL	**0.308**	23.008	1.016	2.070
AMD	**0.269**	16.920	1.146	1.617
AMZN	**0.310**	586.176	10.887	35.085
GOOGL	**0.314**	325.065	7.836	19.530
MSFT	**0.344**	39.834	0.846	1.972
NVDA	**0.364**	24.417	1.203	1.360
PFE	**0.340**	7.113	0.603	0.829
SHOP	**0.385**	203.662	31.288	31.209
WMT	**0.291**	27.182	1.392	0.967

moving average (ARIMA) model (Li et al., 2020), a generalised autoregressive conditional heteroscedasticity (GARCH) model (Li et al., 2020), and a Markov chain Monte Carlo (MCMC) Ornstein–Uhlenbeck (OU) stochastic process (Donado et al., 2017). All these methods are commonly employed in time series financial predictions. The parameters for ARIMA and GARCH were evaluated with the method of moments, while the parameters for the OU process were evaluated with nested sampling (Sivia & Skilling, 2006), a state-of-the-art Bayesian inference technique. Their predictions, as well as those obtained with the proposed quantum-hybrid approach, were compared over a period of thirty trading days with the root mean square value (RMSV) metric. The results are reported in Table 4. QuantumPath outperforms OU, ARIMA and GARCH.

6. Summary and conclusions

This paper proposed a new framework, called QuantumPath, for long-term financial predictions. The expected financial trajectory is predicted with a Feynman–Dirac path integral which results from the stochastic equation associated with the financial instruments. The path integral parameters – the drifts and the volatilities at all times – are predicted with a deep Bayesian network. The Monte Carlo evaluation of the path integral, which involved sampling the data distribution, is performed with a t-GAN which incorporates mechanisms for short and long-term memory as well as causality. Prices for nine stocks, including eight stocks from the S&P 500, were predicted over periods of twenty and thirty days with high precision.

Typically, the temporal evolution of a financial instrument is described with stochastic equations (Linetsky, 1997). In this paper, it has been demonstrated that such an equation may be formulated in terms of a Feynman–Dirac path integral. Quantum computing is not, therefore, introduced *ad hoc*, but is rather the result of the stochastic nature of the stocks: a data-centric approach. These path integrals make it possible to predict the expected value of a financial instrument at a later time. The Monte Carlo evaluation of the path integrals involves sampling the data distribution. In order to optimise the sampling and the precision, and to minimise the computational complexity and the number of trajectories generated, the trajectories are generated with a t-GAN, which means that only the most probable paths are considered. Therefore, the sampling mechanism is learned and not assumed, while being optimal in the Wasserstein sense. The same may be said about the parameters which are learned and predicted with a deep Bayesian neural network rather than being fixed by inspection. In addition to financial predictions, this QuantumPath framework may be applied to risk mitigation, portfolio management, and crisis management. Risk mitigation and portfolio management are closely related. Indeed, our approach allows us to predict stock prices over a given period of time. By knowing the behaviour of the stocks forming a portfolio, it becomes possible to reallocate the assets in order to maximise the return on investment, thus mitigating the risk. As for crisis management, a sudden decline of stock prices may be an indicator of an emerging crisis (Liu et al., 2021; Samimi & Samimi, 2021). By knowing in advance, it is possible to plan a better management strategy. We plan to apply this approach to other financial instruments such as options and forward and future contracts, as well as for forecasting the evolution of the pandemic from the financial market.

CRediT authorship contribution statement

Farzan Soleymani: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Eric Paquet:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

All authors approved the version of the manuscript to be published.

References

Adler, J., & Lunz, S. (2018). Banach wasserstein gan. In *Advances in neural information processing systems* (pp. 6754–6763).
Aizenman, M., Sims, R., & Warzel, S. (2006). Stability of the absolutely continuous spectrum of random Schrödingeroperators on tree graphs. *Probability Theory and Related Fields*, *136*, 363–394.
Amin, A. A. (2019). Kullback-leibler divergence to evaluate posterior sensitivity to different priors for autoregressive time series models. *Communications in Statistics. Simulation and Computation*, *48*, 1277–1291.
Andreev, A. (2006). Schrödingerequation. In *Atomic spectroscopy: introduction to the theory of hyperfine structure* (pp. 13–42).
Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.
Ashraf, B. N. (2020). Stock markets' reaction to covid-19: cases or fatalities? *Research in International Business and Finance*, Article 101249.
Ba, J., & Frey, B. (2013). Adaptive dropout for training deep neural networks. *Advances in Neural Information Processing Systems*, *26*, 3084–3092.
Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.
Bau, D., Zhu, J.-Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., & Torralba, A. (2019). Seeing what a gan cannot generate. In *Proceedings of the ieee international conference on computer vision* (pp. 4502–4511).
Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*, 859–877.
Cai, L., Chen, Y., Cai, N., Cheng, W., & Wang, H. (2020). Utilizing amari-alpha divergence to stabilize the training of generative adversarial networks. *Entropy*, *22*(410).
Cervelló-Royo, R., & Guijarro, F. (2020). Forecasting stock market trend: A comparison of machine learning algorithms. *Finance, Markets and Valuation*, *6*, 37–49.
Chen, M.-H., & Shao, Q.-M. (1997). Performance study of marginal posterior density estimation via kullback-leibler divergence. *Test*, *6*, 321–350.
Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
David, J. M. (1992). A practical bayesian framework for backprop networks. *Neural Computation*.

Deshpande, I., Zhang, Z., & Schwing, A. G. (2018). Generative modeling using the sliced wasserstein distance. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3483–3491).

Dirac, P. A. (2005). The lagrangian in quantum mechanics. In *Feynman's thesis—a new approach to quantum theory* (pp. 111–119). World Scientific.

Donado, F., Moctezuma, R., López-Flores, L., Medina-Noyola, M., & Arauz-Lara, J. (2017). Brownian motion in non-equilibrium systems and the ornstein-uhlenbeck stochastic process. *Scientific Reports*, *7*, 1–7.

Du, D.-Z., & Pardalos, P. M. (2013). *Minimax and applications, Vol. 4*. Springer Science & Business Media.

Dukler, Y., Li, W., Lin, A., & Montúfar, G. (2019). Wasserstein of wasserstein loss for learning generative models. In *International conference on machine learning* (pp. 1716–1725). PMLR.

Esteve, J. (2002). Origin of the anomalies: The modified heisenberg equation. *Physical Review D*, *66*, Article 125013.

Frogner, C., Zhang, C., Mobahi, H., Araya, M., & Poggio, T. A. (2015). Learning with a wasserstein loss. *Advances in Neural Information Processing Systems*, *28*, 2053–2061.

Goldin, G. (2008). Nonlinear quantum mechanics: Results and open questions. *Physics of Atomic Nuclei*, *71*(884).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Habeck, M. (2012). Evaluation of marginal likelihoods via the density of states. In *Artificial intelligence and statistics* (pp. 486–494).

He, P., Sun, Y., Zhang, Y., & Li, T. (2020). Covid–19's impact on stock prices across different sectors—an event study based on the chinese stock market. *Emerging Markets Finance and Trade*, *56*, 2198–2212.

He, H., Wang, H., Lee, G.-H., & Tian, Y. (2019). Probgan: Towards probabilistic gan with theoretical guarantees. In *Iclr (poster)*.

Hernandez-Lobato, J., Li, Y., Rowland, M., Bui, T., Hernández-Lobato, D., & Turner, R. (2016). Black-box alpha divergence minimization. In *International conference on machine learning* (pp. 1511–1520). PMLR.

Hurtado, J., & Barbat, A. H. (1998). Monte carlo techniques in computational stochastic mechanics. *Archives of Computational Methods in Engineering*, *5*, 3–29.

Ingold, G.-L. (2002). Path integrals and their application to dissipative quantum systems. In *Coherent evolution in noisy environments* (pp. 1–53). Springer.

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., & Bennamoun, M. (2020). Hands-on bayesian neural networks–a tutorial for deep learning users. arXiv preprint arXiv:2007.06823.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International conference on machine learning* (pp. 2342–2350). PMLR.

Kakushadze, Z. (2015). Path integral and asset pricing. *Quantitative Finance*, *15*, 1759–1771.

Kingma, D., & Ba, J. (2015). Adam: a method for stochastic optimization 2014. (p. 15). arXiv preprint arXiv:1412.6980.

Kononenko, I. (1989). BayesIan neural networks. *Biological Cybernetics*, *61*, 361–370.

Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *European conference on computer vision* (pp. 47–54). Springer.

Li, T., Zhong, J., & Huang, Z. (2020). Potential dependence of financial cycles between emerging and developed countries: Based on arima-garch copula model. *Emerging Markets Finance and Trade*, *56*, 1237–1250.

Linetsky, V. (1997). The path integral approach to financial modeling and options pricing. *Computational Economics*, *11*, 129–163.

Liu, K. (2018). Varying k-lipschitz constraint for generative adversarial networks. *ArXiv*, arXiv–1803.

Liu, Y., Wei, Y., Wang, Q., & Liu, Y. (2021). International stock market risk contagion during the covid-19 pandemic. *Finance Research Letters*, Article 102145.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., & Bousquet, O. (2018). Are gans created equal? a large-scale study. *Advances in Neural Information Processing Systems*, *31*, 700–709.

Masujima, M. (2008). *Path integral quantization and stochastic quantization*. Springer Science & Business Media.

Mullachery, V., Khera, A., & Husain, A. (2018). BayesIan neural networks. arXiv preprint arXiv:1801.07710.

Nicola, M., Alsafi, Z., Sohrabi, C., Kerwan, A., Al-Jabir, A., Iosifidis, C., Agha, M., & Agha, R. (2020). The socio-economic implications of the coronavirus pandemic (covid-19): A review. *International Journal of Surgery (London, England)*, *78*(185).

Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, *53*, 3007–3057.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. arXiv preprint arXiv:1312.6026.

Perepelitsa, D. V. (2018). Path integrals in quantum mechanics. MIT Department of Physics.

Saatci, Y., & Wilson, A. G. (2017). BayesIan gan. *Advances in Neural Information Processing Systems*, *30*, 3622–3631.

Saito, M., Matsumoto, E., & Saito, S. (2017). Temporal generative adversarial nets with singular value clipping. In *Proceedings of the ieee international conference on computer vision* (pp. 2830–2839).

Samimi, A., & Samimi, M. (2021). Investigating the strategic concept of risk management in the stock market and investing. *Journal of Engineering in Industrial Research*, 2.

Santana, S. R., & Hernández-Lobato, D. (2020). Adversarial α-divergence minimization for bayesian approximate inference. *Neurocomputing*.

Sivia, D., & Skilling, J. (2006). *Data analysis: A bayesian tutorial*. USA: Oxford University Press.

Soleymani, F., & Paquet, E. (2020). Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder-deepbreath. *Expert Systems with Applications*, Article 113456.

Soleymani, Farzan, & Paquet, Eric (2021). Deep graph convolutional reinforcement learning for financial portfolio management-deeppocket. *Expert Systems with Applications*, 115127. http://dx.doi.org/10.1016/j.eswa.2021.115127.

Springenberg, J. T., Klein, A., Falkner, S., & Hutter, F. (2016). BayesIan optimization with robust bayesian neural networks. *Advances in Neural Information Processing Systems*, *29*, 4134–4142.

Strukov, L., & Timan, A. F. (1977). Mathematical expectation of continuous functions of random variables, smoothness and variance. *Siberian Mathematical Journal*, *18*, 469–474.

Thanh-Tung, H., & Tran, T. (2020). Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)* (pp. 1–10). IEEE.

Truman, A. (1978). Some applications of vector space measures to non-relativistic quantum mechanics. In *Vector space measures and applications i* (pp. 418–441). Springer.

Tulczyjew, W. M. (1977). The legendre transformation. In *Annales de l'ihp physique théorique, Vol. 27* (pp. 101–114).

Ulusoy, I., & Bishop, C. M. (2006). Comparison of generative and discriminative techniques for object detection and classification. In *Toward category-level object recognition* (pp. 173–195). Springer.

Wan, R., Mei, S., Wang, J., Liu, M., & Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, *8*(876).

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., & Zhang, D. (2017).

Weng, L. (2019). From gan to wgan. arXiv preprint arXiv:1904.08994.

Wu¨thrich, A. (2010). *The genesis of feynman diagrams* (pp. 51–64). Springer.

Zhang, C., Bu¨ttepage, J., Kjellstr¨om, H., & Mandt, S. (2018). Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*, 2008–2026.

Zhang, D., Hu, M., & Ji, Q. (2020). Financial markets under the global pandemic of covid-19. *Finance Research Letters*, Article 101528.

Zhang, K., Zhong, G., Dong, J., Wang, S., & Wang, Y. (2019). Stock market prediction based on generative adversarial network. *Procedia Computer Science*, *147*, 400–406.

Zhao, J., Mathieu, M., & LeCun, Y. (2016). Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126.

Zhou, X., Pan, Z., Hu, G., Tang, S., & Zhao, C. (2018). Stock market prediction on high-frequency data using generative adversarial nets. *Mathematical Problems in Engineering*, 2018.

Ziemba, W. T., Zhitlukhin, M., & Lleo, S. (2017). *Stock market crashes: Predictable and unpredictable and what to do about them, Vol. 13*. World Scientific.