Computational Economics
https://doi.org/10.1007/s10614-024-10792-1

Neural Network for Valuing Bitcoin Options Under Jump-Diffusion and Market Sentiment Model

Edson Pindza[1] · **Jules Clement**[2] · **Sutene Mwambi**[2] · **Nneka Umeorah**[3]

Accepted: 2 November 2024

Abstract
Cryptocurrencies and Bitcoin, in particular, are prone to wild swings resulting in frequent jumps in prices, making them historically popular for traders to speculate. It is claimed in recent literature that Bitcoin price is influenced by sentiment about the Bitcoin system. Transaction, as well as the popularity, have shown positive evidence as potential drivers of Bitcoin price. This study introduces a bivariate jump-diffusion model to capture the dynamics of Bitcoin prices and the Bitcoin sentiment indicator, integrating trading volumes or Google search trends with Bitcoin price movements. We derive a closed-form solution for the Bitcoin price and the associated Black–Scholes equation for Bitcoin option valuation. The resulting partial differential equation for Bitcoin options is solved using an artificial neural network, and the model is validated with data from highly volatile stocks. We further test the model's robustness across a broad spectrum of parameters, comparing the results to those obtained through Monte Carlo simulations. Our findings demonstrate the model's practical significance in accurately predicting Bitcoin price movements and option values, providing a reliable tool for traders, analysts, and risk managers in the cryptocurrency market.

Keywords Jump-diffusion model · Cryptocurrencies · PDE · Bitcoin · Black–Scholes equation · Artificial neural network

JEL Classification C15 · C45 · C53 · G17

1 Introduction

Sene et al. (2021), Olivares (2020), Chen and Huang (2021), Grohs et al. (2018) Bitcoin, a decentralized network-based digital currency and payment system, is a special type of cryptocurrency developed in 2009 Nakamoto (2008) by a person or a group of persons known under the name of Satoshi Nakamoto. The soar in bitcoin appreciation has been accompanied by high uncertainty and volatility, which

Extended author information available on the last page of the article

Published online: 21 November 2024

surrounds future prices, and this has attracted rapidly increasing research into this digital asset. Policymakers globally are concerned whether bitcoin is decentralized and unregulated and whether it could be a bubble which threatens the stability of a given financial system Cheah and Fry (2015). Regardless of the speculation, traders are still interested in the capacity of Bitcoin to generate more returns and the use of Bitcoin derivatives as a risk management and diversification tool. Blockchain is the underlying technology that powers Bitcoin by recording transactions in a distributive manner, removing alteration and censorship. Following the success of Bitcoin and its growing community, many other alternatives to Bitcoin have emerged. There are more than 5000 tradable cryptocurrencies[1] with a total market capitalization of USD 248 billion at the time of this writing (see Yermack (2015, 2017) for further background on Bitcoin and its technology).

On the one hand, the cryptocurrency market is known to be highly volatile (Dwyer 2015; Katsiampa 2017) due to its sensibility to new information, whether fundamental or speculative since it does not rely on the stabilizing policy of a central bank Cheah and Fry (2015). On the other hand, the relative illiquidity of the market with no official market makers makes it fundamentally fragile to large trading volumes and market imperfections and thus more prone to large swings than other traded assets, [See Scaillet et al. (2017)]. This concept results in frequent jumps of larger amplitude than what a continuous diffusion process can explain. Due to its local Markov property, i.e., the asset price changes only by a small amount in a short interval of time. When analyzing cryptocurrency data, it is interesting to consider processes that allow for random fluctuations that have more than a marginal effect on the cryptocurrency's price. Such a stochastic process that enables us to incorporate this type of effect is the jump process. This process allows the random fluctuations of the asset price to have two components, one consisting of the usual increments of a Wiener process; the second provides for "large" jumps in the asset price from time to time. Shortly after the development of the Black–Scholes option valuation formula, Merton (1976) developed a jump-diffusion model to account for the excess kurtosis and negative skewness observed in log-return data (see, Matsuda (2004a)). This jump process is appended to the Black–Scholes geometric stochastic differential equation.

Several studies have used the daily bitcoin data to document the impact of jump as a crucial feature of the cryptocurrency dynamics. Chaim and Laurini (2018) observed that jumps associated with bitcoin volatility are permanent, whereas the jumps to mean returns are said to have contemporaneous effects. The latter equally capture large price bubbles, mainly negative, and are often associated with hacks and unsuccessful fork attempts in the cryptocurrency markets. Hilliard and Ngo (2022) further investigated the characteristics of bitcoin prices and derived a model which incorporates both jumps and stochastic convenience yield. The result was tested with data obtained from the Deribit exchange, and they observed that modelling bitcoin prices as a jump-diffusion model outperformed the classical Black–Scholes models. Philippas et al. (2019) observed that during periods of high uncertainty, some

[1] https://coinmarketcap.com/.

informative signals, which are proxied by Google search volume and Twitter tweets, have a partial influence on Bitcoin prices and price jumps.

In recent years, the focus has been on identifying the main drivers of Bitcoin price evolution in time. Many researchers have assigned the high volatility in Bitcoin prices to the sentiment and popularity of the Bitcoin system. Though these are not directly observable, they may be considered as indicators from the transaction volumes or the number of Google searches or Wikipedia requests about the topic, see for example, Kristoufek (2013, 2015), Kim et al. (2015) and Bukovina and Marticek (2016). Authors in Bukovina and Marticek (2016) use a Bitcoin sentiment measure from Sentdex.com and develop a discrete-time model to show that excessive confidence in the system may boost a bubble in the Bitcoin system. These sentiment-based data were collected through Natural Language Processing techniques to identify a string of words conveying positive, neutral or negative sentiment on Bitcoin. Furthermore, Cretarola et al. (2017) introduce a bivariate model in continuous time to describe both the dynamics of the Bitcoin sentiment indicator and the corresponding Bitcoin price. By fitting their bivariate model to market data, they consider both the volume and the number of Google searches as proxies for the sentiment factor.

While traditional financial theory relies on assumptions of market efficiency, normally distributed returns, and no-arbitrage, emerging research suggests cryptocurrency markets exhibit different characteristics. These markets appear prone to inefficiencies, fat-tailed non-normal distributions, and frequent arbitrage opportunities according to Kabašinskas and Šutienė (2021). For example, the extreme volatility and relative illiquidity of cryptocurrencies can lead to dislocations between markets where arbitrageurs can profit. Additionally, the return distributions tend to exhibit higher kurtosis and skewness than normal distributions. However, despite violating some traditional assumptions, jump-diffusion models can still provide a useful starting point for modelling cryptocurrency dynamics. The jumps can account for extreme price fluctuations beyond what continuous diffusion alone predicts. While the model may require modifications over time as more data becomes available, it captures key features like volatility clustering and significant outliers. The sentiment indicator variable also represents an initial attempt to incorporate a behavioural factor affecting prices.

Our technique incorporates the one similar to Cretarola et al. (2017), who developed a continuous bivariate model that described the bitcoin price dynamics as one factor and a sentiment indicator as the second factor. We further added a jump-diffusion component to the SDE with the aim of capturing the occurrence of rare or extreme events in the bitcoin price return. Furthermore, we introduce the artificial neural network and propose a trial solution that solves the associated Black–Scholes partial differential equation (PDE) for the Bitcoin call options with European features. This concept is equally different from the univariate jump-diffusion model used, for example, by Chen and Huang (2021). We further implemented the number of Google searches as a Bitcoin sentiment indicator in this paper. This choice is due to their unique transparency in contrast to other social media-driven measures, and they have the tendency to gauge behaviour

instead of searching for it. Therefore, using search-based data as sentiment indices has the potential to reveal the underlying beliefs of populations directly.

In this paper, we present a novel modelling framework using a bivariate jump-diffusion model and sentiment indicator. This framework serves as a foundational tool for pricing and derivatives valuation in cryptocurrency markets. We acknowledge the model's limitations in the context of the evolving understanding of these new markets. As future research expands knowledge of distributional properties, market microstructure issues, and other intricacies, the modelling approach can be enhanced. Nonetheless, our proposed model offers a unique and significant contribution to the field of financial engineering in the cryptocurrency space. The key contributions of this paper are as follows:

- We introduce a bivariate jump-diffusion model to capture the dynamics of the Bitcoin sentiment indicator, which encompasses trading volumes or Google search trends alongside the corresponding Bitcoin price movements.
- We derive a practical and applicable solution by formulating a closed-form solution for the Bitcoin price and the associated Black–Scholes equation for valuing Bitcoin options.
- The resulting Bitcoin option partial differential equation (PDE) is solved using an artificial neural network, with the model validated against data from highly volatile stocks.
- Finally, we test the robustness of the model across a broad spectrum of parameters, comparing the results to those obtained via the Monte Carlo simulation framework.

The practical significance of this research lies in its ability to enhance the understanding and prediction of Bitcoin market dynamics through a sophisticated modelling approach. Incorporating sentiment indicators ensures that the model provides a more comprehensive view of market dynamics. Using a neural network to solve the Bitcoin option PDE demonstrates the integration of advanced machine learning techniques in financial modelling. These network structures can handle non-linearities and complex patterns that traditional methods may struggle with. Testing the model's robustness across a wide range of parameters ensures its reliability and generalizability. Comparing the results with those obtained via Monte Carlo simulations provides a benchmark to validate the model's accuracy. Thus, by leveraging sentiment indicators, formulating the PDE, employing advanced computational techniques for the quasi-analytical scheme, and validating robustness, the research offers valuable tools for traders, analysts, and financial professionals to navigate the volatile cryptocurrency market more effectively.

The rest of the paper is organized as follows: Section 2 introduces the methodology and highlights the strengths and the limitations of the proposed model. Section 3 describes the concept of the artificial neural network, as well as its applications in solving the option pricing differential equations, Sect. 4 discusses the numerical implementation findings, and the last section concludes the work.

2 Methodology

The bivariate jump diffusion and sentiment model is an advanced financial modelling framework that integrates market dynamics and sentiment analysis to capture the complexities of asset price movements more accurately (in this case, bitcoin prices). Sentiment analysis incorporates market sentiment derived from news articles, social media, analyst reports, and other textual data sources. In addition, sentiment scores are quantified and integrated into the model to reflect how positive or negative news impacts market behaviour and can be treated as an external factor influencing the likelihood and magnitude of jumps in modelling asset prices. Thus, by capturing both the inherent randomness of asset prices and the external impacts of sentiment, this model offers enhanced predictive capabilities and practical applications in option pricing. Specifically, the overview of the methodology entails understanding the jump-diffusion model, which is an extension of the Black and Scholes (1973). Jump-diffusion models are continuous-time stochastic processes introduced in quantitative finance by Merton (1973), and they reproduce stylised facts observed in asset price dynamics, such as mean-reversion and jumps. Incorporating jumps into the modelling of asset prices is essential, especially when capturing the dynamics of highly volatile assets such as Bitcoin. There are two ways to expound the description of jump processes.

1. One is to describe the jump in **absolute terms** (this is useful if we are focusing on prices or interest rates),
2. The other is to describe the jump in **proportional terms** (this is more useful if our focus is on returns, as in this study).

In addition to jump-diffusion models, Levy processes appear as an alternative for capturing large deviations in asset prices. Levy processes allow sample paths with frequent discontinuities, enabling them to generate heavy-tailed distributions. Some key examples include variance gamma processes, normal inverse Gaussian processes, and generalized hyperbolic processes. Watana Be (2006) demonstrated that Bitcoin returns exhibited heavy tails and proposed using a variance gamma process to model the price dynamics. Kim et al. (2017) found evidence that variance gamma processes fit cryptocurrency log returns compared to classical diffusion. Levy processes have also been employed to model stochastic volatility in Bitcoin prices (Caporale et al., 2018). Overall, Levy processes provide an alternative class of models with more flexibility to account for extreme deviations. They have shown promise in fitting the empirical distributional properties of cryptocurrency returns. As this work initially focuses on extending jump-diffusion models, exploring Levy processes is a worthwhile direction for future research. However, their ability to capture heavy tails and discontinuities suggests that Levy processes could serve as a valuable modelling technique in this domain.

2.1 Strengths and Limitations of the Proposed Model

The bivariate jump-diffusion model, a unique feature of our proposed model, serves as a crucial first step in applying financial engineering techniques to the emerging field of cryptocurrency. By incorporating stochastic volatility, discrete jumps, and a sentiment indicator, it captures several distinctive features of cryptocurrencies. This preliminary pricing and derivatives valuation framework can be further developed as the market matures. While the model's assumptions will need validation as more data becomes available, it currently provides valuable insights for investment analysis. We recognize the need for continuous refinement as cryptocurrency finance evolves into a distinct field. This paper aims to establish the initial modelling foundation, which can be progressively enhanced as research advances. Potential future improvements include more flexible return distributions, multifactor models to account for additional risks, regime-switching models to reflect market phases, and calibration of the sentiment factor from diverse data sources.

In addition, the current proposed model offers a useful starting point, we recognize this initial framework may require modifications over time to align with market realities. As more cryptocurrency data becomes available, the distributional properties and other intricacies of these markets will become clearer. If the actual return distributions exhibit higher kurtosis than the normal distribution assumed, the model may need to be adjusted accordingly. Furthermore, if inefficiencies and arbitrage opportunities remain prevalent, the dynamics may diverge from a pure jump-diffusion process. As researchers expand their knowledge of the microstructure, behavioural components, and risks in cryptocurrency markets, our modelling approach can be enhanced. In summary, we emphasize this work represents a starting point for modelling cryptocurrencies and enabling financial applications. The proposed bivariate jump-diffusion model and sentiment indicator capture essential dynamics but may require adjustments as knowledge develops. We welcome future research to build on these initial techniques for pricing and valuation of cryptocurrency-denominated assets.

The overall structure is given by

$$dS_t = S_t(\mu_d - \lambda k)dt + \sigma_d S_t dW_t + S_t J_t \mathrm{d}N_t, \qquad S_0 = s_0 \in \mathbb{R}_+ \tag{1}$$

where $S = \{S_t,\ t \geq 0\}$ be the price process of Bitcoin. Let y_t be the absolute jump size, where we assume that the bitcoin price jumps from S_t to $y_t S_t$ in a small time interval $\mathbb{d}t$. The relative price jump size, $J_t = y_t - 1$, is the percentage change in the bitcoin price influenced by the jump. Using the Merton jump-diffusion model, the absolute price jump size is a non-negative random variable which is log-normally distributed, i.e $\ln(y_t) \sim i.i.d\ \mathcal{N}(\mu, \delta^2)$ [See Matsuda, (2004a)]. The other parameters $\mu_d, \sigma_d, k, W_t, \lambda$ and N_t denote the drift rate, volatility, average jump size, Weiner process, jump intensity and the Poisson process, respectively. The next strategy entails incorporating the market sentiment which are evident in the Bitcoin market. This sentiment index may be the volume/the number of Bitcoin transactions or the number of internet searches within a fixed time period.

With that into perspective, consider a probability space $(\Omega, \mathcal{F}, \mathbf{P})$ endowed with a filtration $\mathbb{F} = \{\mathcal{F}_t,\ t \geq 0\}$ that satisfies the usual conditions of right-continuity and completeness, where $\mathcal{F}_t = \sigma\big(W_s, N_s; 0 \leq s \leq t\big)$: W_t is a standard Brownian motion, and N_t denotes a counting process that can be represented as $N_t = \sum_i 1_{\{T_i \leq t\}}$ with a sequence of stopping times $0 < T_1 < T_2 < \cdots < T$ whose number is finite with probability one. Factoring in the sentiment index, we further redefine the dynamics of the Bitcoin price by the following jump-diffusion stochastic differential equation

$$dS_t = S_t P_{t-\tau}(\mu_d - \lambda k)dt + \sigma_d \sqrt{P_{t-\tau}} S_t dW_t + S_t P_{t-\tau}(y_t - 1)dN_t, \qquad S_0 = s_0 \in \mathbb{R}_+ \tag{2}$$

where μ_d is the diffusion mean, σ_d the diffusion volatility, λ the jump intensity rate and $\mathbb{E}[J_t] = \mathrm{e}^{\mu + \frac{\delta^2}{2}} - 1 = k$, the expected proportional jump size. The one-dimensional Poisson process is discontinuous with a constant jump λ and is given by

$$\mathrm{d}N_t = N_{t+dt} - N_t = \begin{cases} 0 \text{ with probability } 1 - \lambda \mathrm{d}t, \\ 1 \text{ with probability } \lambda \mathrm{d}t \end{cases}.$$

Also, the exogenous process $P = \{P_t,\ t \geq 0\}$ is a stochastic factor representing the sentiment index in the Bitcoin market, satisfying

$$dP_t = P_t \mu_p dt + \sigma_p P_t dZ_t, \qquad P_t = \phi(t),\ t \in [-L, 0]. \tag{3}$$

where $\mu_p \in \mathbb{R} - \{0\}$, $\sigma_p \in \mathbb{R}_+$, $L \in \mathbb{R}_+$, $Z = \{Z_t,\ t \geq 0\}$ is a standard $\mathbb{F}$-Brownian motion on $(\Omega, \mathcal{F}, \mathbf{P})$, which is $\mathbf{P}$-independent of W, and $\phi : [-L, 0] \to [0, +\infty)$ is a continuous (deterministic) initial function. The non-negative property of the function ϕ corresponds to the requirement that the minimum level for sentiment is zero. The delay variable $P_{t-\tau}$ accounts for the effect of investor sentiment on volatility, and this is supported by findings that factors like public attention and transaction volume influence cryptocurrency volatility (e.g. (Bukovina and Marticek, 2016; Cretarola et al., 2017)). This variable affects the instantaneous variance of the Bitcoin price modulated by σ_d. The solution to Eq. (3) exists in its closed form Black and Scholes (1973), and P_t is lognormally distributed for each $t > 0$.

The system given by Eqs. 2 and 3 is well-defined in $\mathbb{R}_+$ as stated in the following theorem

Theorem 2.1 *In the market model described previously, the following holds*:

1. *the bivariate stochastic delayed differential equation*

$$\begin{cases} dS_t = S_t P_{t-\tau}(\mu_d - \lambda k)dt + \sigma_d \sqrt{P_{t-\tau}} S_t dW_t + S_t P_{t-\tau}(y_t - 1)dN_t, & S_0 = s_0 \in \mathbb{R}_+ \\ dP_t = P_t \mu_p dt + \sigma_p P_t dZ_t, & P_t = \phi(t),\ t \in [-L, 0]. \end{cases} \tag{4}$$

has a continuous, $\mathbb{F}$-adapted, unique strong solution $(S, P) = \{(S_t, P_t),\ t \geq 0\}$ given by

$$S_t = S_0 \exp\left(\left((\mu_d - \lambda k) - \frac{\sigma_d^2}{2}\right)\int_0^t P_{u-\tau}du + \sigma_d \int_0^t \sqrt{P_{u-\tau}}dW_u + \int_0^t \ln(1 + P_{u-\tau}(y_u - 1))dN_u\right) \tag{5}$$

$$P_t = \phi(0)\exp\left(\left(\mu_d - \frac{\sigma_d^2}{2}\right)t + \sigma_d Z_t\right), \qquad t \geq 0 \tag{6}$$

The solution (5) can be obtained by directly applying the Ito's formula for jump-diffusion processes below.

Proposition 2.2 (Ito's formula for diffusion processes (Tankov, 2003), *Proposition* 8.14, *p*.275) *Let X be a diffusion process with jumps, defined as*

$$X_t = X_0 + \int_0^t b_s \mathrm{d}s + \int_0^t \sigma_s \mathrm{d}W_s + \sum_{i=1}^{N_t} \Delta X_i,$$

where b_t *and* σ_t *are continuous process with*

$$\mathbb{E}\left[\int_0^T \sigma_t^2 \mathrm{d}t\right] < \infty.$$

Then for any $\mathcal{C}^{1,2}$ *function* $f : [0,T] \times \mathbb{R} \Rightarrow \mathbb{R}$, *the process* $Y_t = f(t, X_t)$ *can be represented as*

$$f(t,X_t) - f(0,X_0) = \int_0^t \left[\frac{\partial f}{\partial s}(s,X_s) + \frac{\partial f}{\partial x}(s,X_s)b_s\right]\mathrm{d}s + \frac{1}{2}\int_0^t \sigma_s^2 \frac{\partial^2 f}{\partial x^2}(s,X_s)\mathrm{d}s + \int_0^t \frac{\partial f}{\partial x}(s,X_s)\sigma_s \mathrm{d}W_s + \sum_{i \geq 1, T_i \leq t} [f(X_{T_i-} + \Delta X_i) - f(X_{T_i-})]$$

Following Proposition 2.2, we assume that $f(S_t) = \ln(S_t) \in \mathcal{C}^{1,2}(\mathbb{R})$ and assuming that the process S_t is cáglád. Thus, we have that

$$\ln(S_t) = \ln(S_0) + \int_0^t P_{u-\tau}(\mu_d - \lambda k)du + \frac{1}{2}\int_0^t \sigma_d^2 P_{u-\tau}du + \int_0^t \sigma_d \sqrt{P_{u-\tau}}dW_u + \int_0^t \ln(S_{u^-} + S_u P_{u-\tau}(y_u - 1)) - \ln(S_{u^-})\, dN_u.$$

Proposition 2.3 (Derivation of the Bitcoin option PDE) *Let* $V_t = V(t, S_t)$ *be the price of the derivative that cannot be exercised before maturity. and consider the stock price process described by the Eqs. 2 and 3. Let* $B(t)$ *be the risk-free asset process*:

$$B(t) = e^{rt}, \quad t \in [0, T]$$

where $r > 0$. The price of the European option on a stock S_t under the modified Black–Scholes framework satisfies the following partial differential equation:

$$\frac{\partial V_t}{\partial t} + \mathcal{L}V_t - (\lambda k - \mu_d)S_t P_{t-\tau} = 0, \tag{7}$$

where the differential operator $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{2}\sigma_d^2 P_{t-\tau} S_t^2 \frac{\partial^2}{\partial S^2} + [rS_t + (\mu_d - \lambda k)S_t P_{t-\tau}]\frac{\partial}{\partial S} - r.$$

Proof The Black–Scholes PDE in the Proposition 2.3 above is derived via the hedging argument. Applying Ito's formula,

$$\begin{aligned} dV(t, S_t) = &\left(\frac{\partial V}{\partial t}(t, S_t) + (\mu_d - \lambda k)S_t P_{t-\tau}\frac{\partial V}{\partial S}(t, S_t) + \frac{\sigma_d^2}{2}P_{t-\tau}S_t^2\frac{\partial^2 V}{\partial S^2}(t, S_t)\right)dt \\ &+ \sigma_d\sqrt{P_{t-\tau}}S_t\frac{\partial V}{\partial S}(t, S_t)dW_t + \Delta V(t, S_t)dN_t \end{aligned}$$

We set up a self-financing portfolio X that is comprised of one option and δ amount of the underlying stock, such that the portfolio is riskless, that is insensitive to changes in the price of the security. So, at time t, the value of the portfolio is $X_t = V_t + \delta S_t$. The self-financing assumption implies that

$$dX_t = dV_t + \delta dS_t \tag{8}$$

Substituting the dV_t and dS_t into the Eq. (8), we have

$$\begin{aligned} dX_t = &\left[\left(\frac{\partial V}{\partial t}(t, S_t) + (\mu_d - \lambda k)S_t P_{t-\tau}\frac{\partial V}{\partial S}(t, S_t) + \frac{\sigma_d^2}{2}P_{t-\tau}S_t^2\frac{\partial^2 V}{\partial S^2}(t, S_t)\right)dt\right. \\ &\left. + \sigma_d\sqrt{P_{t-\tau}}S_t\frac{\partial V}{\partial S}(t, S_t)dW_t + \Delta V(t, S_t)dN_t\right] \\ &+ \delta\left[S_t P_{t-\tau}(\mu_d - \lambda k)dt + \sigma_d\sqrt{P_{t-\tau}}S_t dW_t + S_t P_{t-\tau}(y_t - 1)dN_t\right] \end{aligned}$$

So,

$$\begin{aligned} dX_t = &\left(\frac{\partial V}{\partial t}(t, S_t) + (\mu_d - \lambda k)S_t P_{t-\tau}\left(1 + \frac{\partial V}{\partial S}(t, S_t)\right) + \frac{\sigma_d^2}{2}P_{t-\tau}S_t^2\frac{\partial^2 V}{\partial S^2}(t, S_t)\right)dt \\ &+ \sigma_d\sqrt{P_{t-\tau}}S_t\left(\delta + \frac{\partial V}{\partial S}(t, S_t)\right)dW_t + \left(\delta S_t P_{t-\tau}(y_t - 1) + \Delta V(t, S_t)\right)dN_t \end{aligned} \tag{9}$$

On one hand, this portfolio should be riskless and earn a risk-free rate. To be riskless, the second and the third terms involving, respectively, the Brownian motion dW_t and the Poisson process dN_t must be zero. That is

$$\delta = -\frac{\partial V}{\partial S}(t, S_t) \text{ and } \Delta V(t, S_t) = -\delta S_t P_{t-\tau}(y_t - 1)$$

Hence,

$$\Delta V(t, S_t) = S_t P_{t-\tau}(y_t - 1)\frac{\partial V}{\partial S}(t, S_t)$$

Substituting for δ and $\Delta V(t, S_t)$ in Eq. (9) implies that the portfolio follows the process

$$dX_t = \left(\frac{\partial V}{\partial t}(t, S_t) + (\mu_d - \lambda k)S_t P_{t-\tau}\left(1 + \frac{\partial V}{\partial S}(t, S_t)\right) + \frac{\sigma_d^2}{2}P_{t-\tau}S_t^2\frac{\partial^2 V}{\partial S^2}(t, S_t)\right)dt \tag{10}$$

On the other hand, the portfolio must earn the risk-free rate, that is

$$dX_t = rX_t dt$$
$$\left(\frac{\partial V_t}{\partial t} + (\mu_d - \lambda k)S_t P_{t-\tau}\left(1 + \frac{\partial V_t}{\partial S}\right) + \frac{\sigma_d^2}{2}P_{t-\tau}S_t^2\frac{\partial^2 V_t}{\partial S^2}\right)dt = r\left(V_t - \frac{\partial V_t}{\partial S}S_t\right)dt$$

Multiplying both sides by $\frac{1}{dt}$ and re-arranging yields the following Black–Scholes PDE for an option V

$$\frac{\partial V_t}{\partial t} + \frac{\sigma_d^2}{2}P_{t-\tau}S_t^2\frac{\partial^2 V_t}{\partial S^2} + rS_t\frac{\partial V_t}{\partial S} - rV_t = -(\mu_d - \lambda k)S_t P_{t-\tau}\left(1 + \frac{\partial V_t}{\partial S}\right) \tag{11}$$

Finally, re-arranging the equation and recognizing the operator $\mathcal{L}$ acting on V_t

$$\mathcal{L}V_t = \frac{1}{2}\sigma_d^2 P_{t-\tau}S_t^2\frac{\partial^2 V_t}{\partial S^2} + [rS_t + (\mu_d - \lambda k)S_t P_{t-\tau}]\frac{\partial V_t}{\partial S} - rV_t.$$

gives the final PDE below:

$$\frac{\partial V_t}{\partial t} + \mathcal{L}V_t - (\lambda k - \mu_d)S_t P_{t-\tau} = 0.$$

Remark 2.4 This extended Black–Scholes PDE poses challenges for analytical solutions due to the unbounded upside of the European call payoff. Therefore, numerical methods are often used to approximate the solution. Two standard techniques are finite difference methods and Monte Carlo simulation. Finite difference methods involve discretizing the price domain and time dimensions and replacing the derivatives with finite difference approximations. This transforms the PDE into a system of algebraic equations that can be solved recursively. Various finite difference schemes can be employed, such as explicit, implicit, and Crank-Nicolson. Monte Carlo simulation generates sample paths of the underlying price using the assumed stochastic process. The discounted payoffs from each sample path are averaged to estimate the option price. Variance reduction techniques like antithetic sampling and control variates can improve efficiency. Both methods have tradeoffs regarding accuracy, speed,

and implementation complexity. Our proposed neural network approach serves as an alternative way to numerically solve the pricing PDE without discretizing the domain. This offers benefits in terms of generalization and scaling.

3 Neural Network Methodology

The neural network (NN) algorithm is used for solving problems relating to dimensionality reduction (Sarveniazi, 2014; Teli, 2007), visualization of data (Marghescu, 2007), clustering (Du, 2010; Xu & Wunsch, 2005) and classification issues (Nazemi & Dehghan, 2015; O'Dea et al., 2001). It is also used in regression (Bataineh & Marler, 2017; Jiang et al., 2022; Setiono & Thong, 2004) in solving regression problems since they do not require prior knowledge about the distribution of the data. The NN algorithm often has the tendency to predict with higher accuracy than other techniques due to the NN's capability to fit any continuous functions (Hornik, 1991). Mathematically, the NN can be referred to as a directed graph having neurons as vertices and links as edges. Different forms of NN depend on the connectivity of their neurons, and the simplest one is the Multi-layer perceptron, also known as the feedforward NN. Basically, a NN can have only one input and one output layer in its simplest form, and this can be written as:

$$y_k = \Phi\left(b_k + \sum_{i=1}^{m} w_{i,k}\boldsymbol{x}_i\right),$$

where m is the number of input variables, Φ is the activation function, $w_{i,k}$ is the weight of the input layer i with respect to the output k, b is the bias, and the input vector $\boldsymbol{x}$ is connected to the output k (denoting the kth neuron in the output layer) through a biased weighted sum. In the presence of hidden layers positioned between the input layers and the output layers, the output can be written as:

$$y_p = \Phi_{out}\left[b_p^{(2)} + \sum_{k=1}^{m_2} w_{k,p}^{(2)} \times \Phi\left(b_k^{(1)} + \sum_{i=1}^{m_1} w_{i,k}^{(1)}\,\boldsymbol{x}_i\right)\right],$$

where m_1 is the number of input variables, m_2 is the number of nodes in the hidden layer, $\Phi : \mathbb{R} \to \mathbb{R}$ refers to non-linear activation functions for each of the layers in the brackets, and Φ_{out} is the possibly new output function.

Recently, the NN has become an indispensable tool for learning the solutions of differential equations, and this section will utilize the potential of the NN in solving PDE-related problems (Aarts & Van Der Veer, 2001; Dissanayake & Phan-Thien, 1994; Hussian & Suhhiem, 2015; Umeorah & Mba, 2022). Also, the NN has been implemented in solving equations without analytical solutions. For instance, Nwankwo et al. (2023) considered the solution of the American options PDE by incorporating the Landau transformation and solving the corresponding transformed function via a neural network approach. In many scientific

and industrial contexts, there is a need to solve parametric PDEs using NN techniques and Khoo et al. (2021) developed such a technique which solves PDE with inhomogeneous coefficient fields. For high-dimensional parametric PDE, Glau and Wunderlich (2022) analyzed the deep parametric PDE method to solve high-dimensional parametric PDEs with much emphasis on financial applications.

In the following, we describe the Bitcoin options with the corresponding Black–Scholes pricing PDE:

$$\frac{\partial V}{\partial t} + \frac{\sigma^2 S^2}{2} P_{t-\tau} \frac{\partial^2 V}{\partial S^2} + (r + (\mu - \lambda K) P_{t-\tau}) S \frac{\partial V}{\partial S} - rV + (\mu - \lambda K) S P_{t-\tau} = 0 \tag{12}$$

The above equation which is a non-homogeneous PDE can be re-written as

$$\frac{\partial V}{\partial t} + \frac{\sigma_*^2 S^2}{2} \frac{\partial^2 V}{\partial S^2} + \eta S \frac{\partial V}{\partial S} - rV = \beta(S), \tag{13}$$

where $\sigma_*^2 = \sigma^2 P_{t-\tau}$, $\eta = r + (\mu - \lambda K) P_{t-\tau}$, $\beta(S) = -(\mu - \lambda K) S P_{t-\tau}$ and $P_t = \phi(0) \exp\{(\mu_p - \frac{\sigma_p^2}{2})t + \sigma_p Z_t\}$.

Equation (13) can be written in its operator format. Let $\Omega = S_{max}$ and $\mathcal{A}$ be the infinitesimal operator for the stochastic process defined by

$$\mathcal{A} = S\eta(t,s) \frac{\partial}{\partial S} + \frac{S^2 \sigma_*^2(t,S)}{2} \frac{\partial^2}{\partial S^2}, \tag{14}$$

with both the boundary and the terminal value problem written as

$$\begin{aligned} (\partial_t + \mathcal{A} - r) V(t,S) &= \beta(t,S), && t, S \in [0,T] \times \Omega \\ V(T,S) &= g_1(T,S) = \text{Payoff}, && t, S \in [0,T] \times \partial\Omega \\ V(t,S) &= g_2(t,S) = S_t - K\mathrm{e}^{-r(T-t)}, && S \in \Omega \\ V(t,0) &= V_0(t) = 0, && t \in [0,T]\,. \end{aligned} \tag{15}$$

Standard theorems guarantee a classical smooth solution exists for the pricing PDE (3.13) given the assumed dynamics. The continuity and linear growth conditions on the coefficient functions ensure they satisfy Lipschitz continuity. Contingent claims theory shows that Lipschitz continuity is sufficient for existence and uniqueness under the stochastic process specifications. In particular, the smooth past price dependence in the diffusion coefficient allows the application results for delayed Black–Scholes equations. Therefore, the Cauchy problem is well-posed under the model assumptions, and a smooth pricing solution is guaranteed to exist based on the theorems. This provides the theoretical foundation for using a neural network to approximate the price function numerically. Thus, employing the ANN to solve the PDE, we introduce an approximating function $\zeta(t, S : \boldsymbol{\theta})$ with parameter set $\boldsymbol{\theta}$. The loss or cost function associated with this training is measured by how well the approximation function satisfies the differential operator, boundary conditions and the terminal condition of the option pricing PDE. These are given respectively as

- Differential operator $\left\|\partial_t + \mathcal{A} - r)\zeta(t, S : \boldsymbol{\theta})\right\|_{Y_1}^2$, where $Y_1 = [0, T] \times \Omega, v_1$.
- Terminal condition $\left\|\zeta(T, S : \boldsymbol{\theta}) - g_1(T, S)\right\|_{Y_2}^2$, where $Y_2 = \Omega, v_2$.
- Boundary condition $\left\|\zeta(t, 0 : \boldsymbol{\theta}) - V_0(t)\right\|_{Y_3}^2$, where $Y_3 = [0, T], v_3$.
- Boundary condition $\left\|\zeta(t, S : \boldsymbol{\theta}) - g_2(t, S)\right\|_{Y_4}^2$, where $Y_4 = [0, T] \times \partial\Omega, v_4$.

In all these four terms above, the observed error is measured in the Hilbert space L^2-norm, meaning that $\|\lambda(x)\|_{m,n}^2 = \int_m |\lambda(x)|^2 n(x)\mathrm{d}x$ with $n(x)$ being the probability density function which describes the region m. The combination of these terms gives the associated cost or loss function connected with the NN training. Thus, the objective function can be written as:

$$\mathcal{L}(\zeta) = \left\|\partial_t + \mathcal{A} - r)\zeta(t, S : \boldsymbol{\theta})\right\|_{Y_1}^2 + \left\|\zeta(T, S : \boldsymbol{\theta}) - g_1(T, S)\right\|_{Y_2}^2 + \left\|\zeta(t, 0 : \boldsymbol{\theta}) - V_0(t)\right\|_{Y_3}^2 + \left\|\zeta(t, S : \boldsymbol{\theta}) - g_2(t, S)\right\|_{Y_4}^2. \tag{16}$$

Suppose $\mathcal{L}(\zeta) = 0$, then $\zeta(t, S : \boldsymbol{\theta})$ is a solution of the PDE in Eq. (15). The major aim is to obtain a set of parameters $\boldsymbol{\theta}$ such that the function $\zeta(t, S : \boldsymbol{\theta})$ minimizes the observed error of $\mathcal{L}(\zeta)$. The procedure for seeking a good parameter set by minimizing this loss function using the stochastic gradient descent (SGD) optimizer is called "training". Thus, using a Machine Learning approach, and in this case, the artificial NN, it will be feasible to minimize the function $\mathcal{L}(\zeta)$ by applying the SGD approach on the sequence of asset and time points which are drawn at random. A part of the input samples, fully dependent on the mini-batch size, is drawn within each iteration to estimate the gradient of the given objective function. The gradient is then estimated over these mini-batches due to the limitations of the computer memory (Liu et al. (2019).

The training of NN is classified into first identifying the network's hyperparameters, that is, the architectural structure and the loss function of the network. Next, use the SGD to optimize or estimate the loss minimizer, and then the derivatives of the loss function are computed using the backpropagation techniques. Essentially, the process of searching and identifying the best $\boldsymbol{\theta}$ parameter by minimizing the loss function using the gradient descent-based optimizers is referred to as "training". The whole procedure is well illustrated in the Algorithm 1 (Sirignano and Spiliopoulos 2018), and can be implemented in solving the PDE in Eq. (15).

It must also be noted that the gradient descent steps $\nabla_\theta \mathcal{G}(\theta_n; x_n)$ are the unbiased estimates of $\nabla_\theta \mathcal{L}(\zeta(.; \theta_n))$, such that

$$\mathbb{E}[\nabla_\theta \mathcal{G}(\theta_n; x_n)|\theta_n] = \nabla_\theta \mathcal{L}(\zeta(.; \theta_n)),$$

and also a negative correlation exists between the learning rate α_n and n.

The learning rate α is employed to scale the intensity of the parameter updates during the gradient descent. Choosing the learning rate, as the descent parameter α, in the network training is crucial since this factor plays a significant role in aiding the algorithm's convergence to the true solution in the gradient descent. It equally affects the pace at which the algorithm learns and whether or not the cost function is

minimized. When α implies divergence, and thus, the optimal point can be missed. Also, a fixed value of α can most likely make the local optima overshoot after some iterations, leading to divergence. Defining the learning rate as a dynamic decreasing function is preferable since it allows the algorithm to identify the needed point rapidly. Another limitation in employing the gradient descent method is obtaining the value of the initial parameters of the loss function. If the value is significantly close to the local optimum, then the slope of the loss function reduces, thereby leading to the optimal convergence. Otherwise, there will be no convergence as the solution explodes abnormally.

Algorithm 1 Implementing NN in solving PDE

(1) Set up the initial parameter set $\boldsymbol{\theta}_0 = (\boldsymbol{W}_0, \boldsymbol{b}_0)$ as well as the learning rate α_n, where $\boldsymbol{W}_0$ and $\boldsymbol{b}_0$ represent the weight and the bias respectively.

(2) Generate samples drawn randomly from the interior of the domain and from the time/asset boundaries. That is

- Generate (t_n, S_n) from $[0, T] \times \Omega$ in accordance to the probability density v_1.
- Generate ω_n from Ω in accordance to the probability density v_2.
- Generate r_n from $[0, T]$ in accordance to the probability density v_3.
- Generate (τ_n, z_n) from $[0, T] \times \partial\Omega$ in accordance to the probability density v_4.

(3) Estimate the value of squared error $\mathcal{G}(\theta_n; x_n)$ where x_n is a set of randomly sampled points denoted by $x_n = \{(t_n, S_n), \omega_n, r_n, (\tau_n, z_n)\}$. That is, compute the following:

- Compute $\mathcal{G}_1(\theta_n; t_n, S_n) = ((\partial_t + \mathcal{A} - r)\zeta(\theta_n; t_n, S_n))^2$.
- Compute $\mathcal{G}_2(\theta_n; \omega_n) = (\zeta(T, \omega_n) - g_1(T, \omega_n))^2$.
- Compute $\mathcal{G}_3(\theta_n; r_n) = (\zeta(r_n, 0) - V_0(r_n))^2$.
- Compute $\mathcal{G}_4(\theta_n; \tau_n, z_n) = (\zeta(\tau, z_n) - g_2(\tau_n, z_n))^2$.
- Set $\mathcal{G}(\theta_n; x_n) = \mathcal{G}_1(\theta_n; t_n, S_n) + \mathcal{G}_2(\theta_n; \omega_n) + \mathcal{G}_3(\theta_n; r_n) + \mathcal{G}_4(\theta_n; \tau_n, z_n)$.

(4) Compute the gradient $\nabla_\theta \mathcal{G}(\theta_n; x_n)$ using backpropagation techniques.

(5) Take an SGD step at the random points x_n using the learning rates from the Adam optimization method. Use the estimated gradient to update the parameter $\boldsymbol{\theta}_n$. Therefore, in each iteration n, the parameters $\boldsymbol{\theta}_n$ are updated in accordance to the relationship below:

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta \mathcal{G}(\theta_n; x_n) \tag{17}$$

Equation (3.17) can be explicitly written in terms of the weights and bias as

$$\begin{cases} \boldsymbol{W}_{i+1} \leftarrow \boldsymbol{W}_i - \alpha_i \frac{\partial \mathcal{G}(\theta_i; x_i)}{\partial \boldsymbol{W}} \\ \\ \boldsymbol{b}_{i+1} \leftarrow \boldsymbol{b}_i - \alpha_i \frac{\partial \mathcal{G}(\theta_i; x_i)}{\partial \boldsymbol{b}} \end{cases}, \tag{18}$$

where $i = 0, 1, \cdots, n$; and n is the number of training iterations.

(6) Repeat the procedure from (2-4) till the convergence property is satisfied, that is till $\|\theta_{n+1} - \theta_n\|$ becomes very small.

3.1 Solution of Bitcoin Option Pricing PDE

While the pricing PDE has a proven smooth solution under the model dynamics, obtaining the analytical form is intractable. Numerical methods must be used to approximate the solution. Neural networks provide a flexible parametric approach based on their universal approximation theoretical results. Given sufficient network capacity, neural networks can represent a wide class of functions, including solutions to PDEs like the pricing equation. However, challenges remain in finding the optimal network parameters to recover the true solution robustly. While existence is guaranteed, the non-convex optimization of neural networks does not assure convergence to global optimality. Care must be taken in specifying the network architecture, loss function, regularization, and training methodology to promote the learning of the pricing function. Subject to these caveats, neural networks present a promising computational technique for approximating smooth pricing solutions, circumventing discretization of the domain. We note the limitations of using neural networks as function approximators. However, their generalization capabilities provide a methodology for data-driven extraction of the pricing mapping across the entire domain. This avoids the constraints of methods like grid-based techniques that rely on local consistency. Future work should explore neural network training enhancements and theoretical guarantees to ensure robust solutions.

Thus, from Eq. (13), let V be the price of the bitcoin call option with strike price E and S' be the price of the bitcoin option at time t'. Let r be the risk-free rate; σ_*, the volatility of the bitcoin; T, the expiration of the contract and denote bitcoin call price $V = V(t', S')$ to be a function of the remaining time to maturity and the bitcoin price. Considering the European call option, the terminal or the final condition and the boundary conditions are denoted respectively by:

$$\begin{cases} TC : V(T, S') = \max\{S' - E, 0\} \\ \\ BC : V(t', 0) = 0 \\ BC : \frac{V(t', S')}{S'} \to 1 \qquad \text{for} \quad S' \to \infty, \end{cases} \tag{19}$$

for $t' \in [0, T]$ and $S' \in [0, S_{\max}]$, where $S_{\max}$ is the upper limit[2] which the bitcoin can accumulate prior to the option's expiration. With the change of variables $t = \frac{T - t'}{T}$, and $S = \frac{S'}{S_{\max}}$, such that $t \in [1, 0]$ and $S \in [0, 1]$, we can substitute $-T\partial t = \partial t'$ into the PDE (13). This restriction is imposed such that the terminal condition (TC) in Eq. (19) can be transformed into an initial condition (IC). Thus, the PDE reduces to

$$\frac{1}{T}\frac{\partial V}{\partial t} - \frac{\sigma_*^2 S^2}{2}\frac{\partial^2 V}{\partial S^2} - \eta S\frac{\partial V}{\partial S} + rV + \beta(S) = 0, \tag{20}$$

[2] We note that since crypto markets are not regulated, the prices may deviate very largely and the maximum could be infinity. However, since we are considering option pricing with European features, we chose the maximum price that the bitcoin has attained from our dataset.

with the following transformed conditions:

$$\begin{cases} IC : V(0,S) = \max\left\{S - \frac{E}{S_{\max}}, 0\right\}, \quad \forall S \in [0,1] \\ BC : V(t,0) = 0, \quad \forall t \in [1,0] \\ BC : V(t,1) = 1 - \frac{E}{S_{\max}}. \end{cases} \tag{19}$$

Solving the PDE in Eq. (20) using the conditions in Eq. (19) entails introducing a trial solution $\zeta(t, S : \theta)$ which employs the feedforward NN with parameter θ corresponding to the weight and biases of the network's architecture. The trial function $\zeta(t, S : \theta)$ of the NN is constructed such that it satisfies the boundary and the initial conditions Eskiizmirliler et al. (2020); Lagaris et al. (1998); Yadav et al. (2015). This can be written as

$$\zeta(t, S : \theta) = A(t, S) + B(t, S)N(t, S : \theta), \tag{20}$$

where $B(t, S) = tS(1 - S)$. The first term $A(t, S)$ satisfies the boundary and the initial conditions has no adjustable parameters and is denoted by

$$A(t,S) = (1-t)\max\left\{S - \frac{E}{S_{\max}}, 0\right\} + tS\left(1 - \frac{E}{S_{\max}}\right).$$

The last term in Eq. (20) with the NN function has adjustable parameters, and it handles the minimization problem. This term does not contribute to the boundary condition because at boundaries $S = 0, S = 1$ and $t = 0$; the term became exactly zero. Thus, the trial function is explicitly given as

$$\zeta(t, S : \theta) = (1-t)\max\left\{S - \frac{E}{S_{\max}}, 0\right\} + tS\left(1 - \frac{E}{S_{\max}}\right) + tS(1-S)N(t, S : \theta). \tag{21}$$

Given the input values (t, S), the network's output is the net function which is defined by

$$N(t, S : \theta) = \sum_{r=1}^{M} q_r f(w_r t + \gamma_r S + b_r), \tag{22}$$

where M is the total number of the neurons which are present in the hidden layer of the NN; $f(z_r)$ is the activation function; q_r is the synaptic weight of the rth hidden neuron to the output; b_r is the bias value of the rth hidden neuron; w_r and γ_r are the synaptic coefficients from the time input to the rth hidden neuron and from the spatial inputs to the rth hidden neuron respectively.

Finally, with regards to the cost function, we first discretise the domains to convert the PDE in Eq. (20) which are subject to the boundary conditions, into an unconstrained minimization problem. For $S \in [0, 1]$ and $t \in [1, 0]$, consider ΔS to be a uniform step size for the bitcoin asset S, and Δt to be the step size for the time t, such that

$$S_i = i\Delta S \implies \Delta S = \frac{1}{N_s}, \quad \text{for} \quad i = 0, 1, \ldots, N_s \quad \text{and}$$
$$t_j = j\Delta t \implies \Delta t = \frac{1}{N_t}, \quad \text{for} \quad j = 0, 1, \ldots, N_t,$$

where N_s and N_t are the step sizes for the bitcoin asset and the time, respectively. The dataset that will be generated from this discretised domain will consist of a matrix of size $(N_s \times N_t) \times 2$, that is, two columns made up by S_i and t_j, as well as $(N_s \times N_t)$ number of rows. The dataset will be split into the train data and the test dataset and the NN will be trained on the train dataset and tested on a separate dataset.

Using the trial function in Eq. (21), denote the bitcoin option value at time t_j for the asset S_i given by $\zeta(t_j, S_i : \boldsymbol{\theta}) = \zeta(j\Delta t, i\Delta S : \boldsymbol{\theta})$ as $\zeta_{j,i}$, then the cost function which the ANN has to minimize is

$$L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{j=1}^{N_t}\sum_{i=1}^{N_s}\left[\frac{\partial \zeta_{j,i}}{\partial t_j} - \frac{\sigma_*^2 S_i^2}{2}\frac{\partial^2 \zeta_{j,i}}{\partial S_i^2} - \eta S_i \frac{\partial \zeta_{j,i}}{\partial S_i} + r\zeta_{j,i} + \beta(S_i)\right]^2, \tag{23}$$

for $j = 0, 1, \ldots, N_t$ and $i = 0, 1, \cdots, N_s$.

Essentially, the loss function is minimized during the training phase to determine the optimal NN parameters. The partial derivatives in Eq. (23) are computed as follows:

$$\frac{\partial \zeta}{\partial t} = -\max\left\{S - \frac{E}{S_{\max}}, 0\right\} + S\left(1 - \frac{E}{S_{\max}}\right) + tS(1-S)\frac{\partial N(t, S : \boldsymbol{\theta})}{\partial t} + S(1-S)N(t, S : \boldsymbol{\theta}) \tag{24}$$

$$\frac{\partial \zeta}{\partial S} = t\left(1 - \frac{E}{S_{\max}}\right) + tS(1-S)\frac{\partial N(t, S : \boldsymbol{\theta})}{\partial S} + N(t, S : \boldsymbol{\theta})(t - 2tS), \quad \text{for } S \le \frac{E}{S_{\max}} \tag{25}$$

$$\frac{\partial \zeta}{\partial S} = (1-t) + t\left(1 - \frac{E}{S_{\max}}\right) + tS(1-S)\frac{\partial N(t, S : \boldsymbol{\theta})}{\partial S} + N(t, S : \boldsymbol{\theta})(t - 2tS), \quad \text{for } S > \frac{E}{S_{\max}} \tag{26}$$

$$\frac{\partial^2 \zeta}{\partial S^2} = tS(1-S)\frac{\partial^2 N(t, S : \boldsymbol{\theta})}{\partial S^2} + 2t(1-2S)\frac{\partial N(t, S : \boldsymbol{\theta})}{\partial S} + -2tN(t, S : \boldsymbol{\theta}). \tag{27}$$

The partial derivatives of the NN with respect to t, S and S^2 from Eqs. (24–27) are dependent on the nature of the activation function used. For instance, if we consider the sigmoid activation function, then $f(z_r) = (1 + \mathrm{e}^{-z_r})^{-1}$. Then from Eq. (22), $z_r = w_r t + \gamma_r S + b_r$, and thus, the NN function in expanded form can be written as

$$N(t, S : \boldsymbol{\theta}) = \sum_{r=1}^{M} q_r\left(1 + \mathrm{e}^{-(w_r t + \gamma_r S + b_r)}\right)^{-1}, \tag{28}$$

Taking the derivatives of Eq. (28) with respect to t, S and S^2, we have the following

$$\frac{\partial N(t,S:\boldsymbol{\theta})}{\partial t} = \sum_{r=1}^{M} q_r w_r f(z_r)(1-f(z_r)) \tag{29}$$

$$\frac{\partial N(t,S:\boldsymbol{\theta})}{\partial S} = \sum_{r=1}^{M} q_r \gamma_r f(z_r)(1-f(z_r)) \tag{30}$$

$$\frac{\partial^2 N(t,S:\boldsymbol{\theta})}{\partial S^2} = \sum_{r=1}^{M} q_r \gamma_r^2 f(z_r)(1-f(z_r))(1-2f(z_r))\,. \tag{31}$$

where $z_r = w_r t + \gamma_r S + b_r$. For the tanh activation function, $f(z_r) = (1-\mathrm{e}^{-2z_r})(1+\mathrm{e}^{-2z_r})^{-1}$, and the derivatives of the NN are also easily obtained mathematically. However, when the ReLU[3] activation $f(z_r) = \max\{0, z_r\}$ is used, the mathematical NN derivatives become complicated due to the Heaviside step function and the presence of dirac delta in the first and second derivatives, respectively. In the context of backpropagation of error, these derivatives are needed when the weights of the NN are constantly updated. The only assumption is the value zero which is obtained when the derivative at point zero is taken.

4 Numerical Results, Implementation and Discussion

This section introduces the empirical and analytical structure, approximation of parameters, the estimation of option prices, as well as the limitations of the efficiency and no-arbitrage assumptions.

4.1 Empirical Analysis

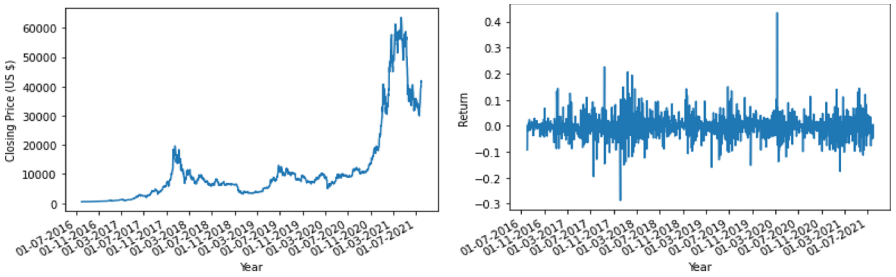
4.1.1 Data Source

For the cryptocurrency data source, we used the bitcoin historical closing prices on the CoinGecko website[4] based on US dollars, covering five years from August 1, 2016, till July 31, 2021. As of access, the global crypto market capitalization of $2.09 Trillion and the CoinGecko tracks 8,934 cryptocurrencies, with 42.2% dominance for bitcoin. Furthermore, we used the Google trend data[5] to extract the bitcoin sentiment data. The Google trend data provides a scaled time series of the number of times bitcoin has been searched, so the maximum is 100. Figures 1 and 2 describe the dataset for the bitcoin prices and the corresponding sentiment trends, respectively.

[3] The tensorflow sigmoid and the ReLU function were used in this research.

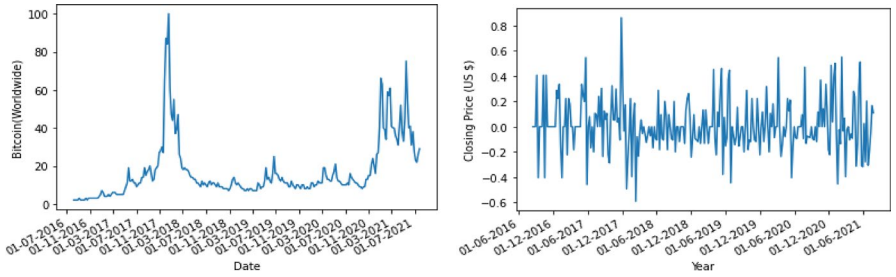
[4] CoinGecko provides a fundamental analysis of the structure of the digital currency market; The data was accessed on August 16, 2021, at https://www.coingecko.com/en.

[5] https://trends.google.com/trends/explore?q=bitcoin.



(a) Bitcoin closing prices (b) Bitcoin returns

Fig. 1 Bitcoin daily closing prices and log returns during 2016–2021



(a) Values (b) Returns

Fig. 2 Sentiment data for bitcoin during 2016–2021

4.1.2 Descriptive Statistics

The dataset is sampled on a daily frequency, and the results are plotted in Fig. 1. We used the adjusted bitcoin closing prices to estimate the continuously compounded returns. For the log-return r_i at time t_i, we used the expression $r_i = \log\left(\frac{S_i}{S_{i-1}}\right)$, where S_i is the bitcoin price at time t_i. Since bitcoin is traded daily, we use the daily sample data, giving 365 observations per year, whereas the trend data is sampled weekly. The descriptive statistics of the dataset used in this work are presented in Table 1.

Both bitcoin prices and the corresponding log-returns react to big events in the cryptocurrency market. From Fig. 1a, a considerable surge in bitcoin prices was observed after March 2017, owing to the widespread interest in cryptocurrencies. This jump was later affected by a series of political interventions leading to a drop in June 2017 [Hou et al. (2020). Furthermore, in late 2020 and early 2021, another dramatic increase in the prices was seen due to the increased acquisition by large investors, financial institutions and corporations. These intensive movements in the cryptocurrency markets have been captured extensively by the corresponding bitcoin sentiment data as found in Fig. 2a.

Table 1 presents the descriptive statistics for the log-returns on the bitcoin closing prices, as well as the sentiment data values. The dual-dataset consists of a sample of 260 weekly sentiment values and 1826 Bitcoin daily closing prices. The Skewness/

Table 1 Descriptive statistics for the log-returns

Statistic	Bitcoin closing prices	Sentiment values
No. of observations	1825	259
Mean	− 0.00241	0.01033
Minimum	− 0.28701	− 0.59205
Q1	− 0.01969	− 0.10536
Median	− 0.00251	0.00000
Q3	0.01336	0.10728
Maximum	0.43371	0.86305
Standard deviation	0.04132	0.20934
Skewness	0.67119	0.45648
Kurtosis	10.53706	1.20553

Kurtosis test is one of three common normality tests designed to detect all the deviations from normality and determine the shape of the distribution for the dataset. For a normal distribution, the skewness is zero, and the kurtosis is three. From the table, the log-return for the dataset of the sentiment and the bitcoin closing prices are positively and highly skewed to the right. With the kurtosis of 1.20553 and 10.53706, the two datasets have heavier, longer and fatter tails than the normal distribution, and they can be referred to as leptokurtic.

4.1.3 Parameter Estimation

In this section, we estimate the parameters for the numerical computation, and the results finally presented are the values of the bitcoin options, having the European call features. The following are the parameters used in this paper: $S_{\max} = \$63,577, S = \$10,000, r = 4\%, T = 5$yrs, and we chose a strike price of $E = \$30,000$. We used the mean and variance of the log-return from the sentiment index data as $\mu_p = 0.01033$ and $\sigma_p = 0.20934$, respectively. Also, from the log-return of the bitcoin closing prices, we calculated the mean and the variance of the log-return as $\mu_d = -0.00241$ and $\sigma_d = 0.04132$, respectively. Next, we estimate the λ, the jump intensity rate, together with k, the expectation of the relative price of the jump size, since it is essential to decide when a jump occurs. This parameter estimation was done using the maximum likelihood estimation method since there are no closed-form expressions for the optimal values of these parameters. Also, the daily bitcoin price return is measured in years, that is, $\Delta t \sim dt = 1/365 = 0.00274$.

Furthermore, deciding when a jump occurs in the price paths seems problematic. We adopt the techniques of Tang (2018) and Hanson and Zhu (2004) to estimate the parameters of the jump-diffusion models, who suggested a specific threshold ϵ with the aim of determining whether a jump has occurred or not. In this case, maximum likelihood estimation is not strongly dependent on the value of the threshold ϵ (Tang, 2018). Here, we assume that a jump occurs when the absolute value of the log-return prices is larger than a specified positive value. The intensity rate λ is measured as

$$\lambda = \frac{\text{number of jumps}}{\text{period length in years}} = \text{number of jumps per year}$$

For our estimate, we set the threshold level $\epsilon = 0.07$. If ϵ is too small, then the majority of the price movements would be considered jumps. On the other hand, if ϵ is large, then the set of absolute jump size y_t could be empty, thereby making the parameters to be estimated to be undetermined. Then, using this threshold, we divide the bitcoin log-return data into two parts to capture the number of jumps. The first part captures the values when the absolute value of the log-return is greater than ϵ, and we assume that a jump occurred here. The other part consists of no jump, and it captures the values whose log-return is lesser than ϵ. From the techniques, we obtained the remaining parameters $\lambda = 31.8$ and $k = \mathbb{E}[J_t] = -0.002195$.

4.2 Numerical Implementation

For the NN architecture, we employed the random search method to obtain the optimal hyperparameter. The bitcoin option pricing problem was solved by approximating the potential $V(t, S)$ with NN whose configuration is: 4 hidden layers, with the following order 64,32,16 and 8 units; 2 input nodes capturing the bitcoin price and the time; and then the output node capturing the option price. (Hence, the configuration 2-64-32-16-8-1). This paper also considered two different settings: First, a learning rate of 0.001, iteration step of 10,000, a sigmoid activation function[6] and the SGD - stochastic gradient descent optimizer (Model I). Secondly, a learning rate of 0.001, iteration step of 10,000, a ReLU activation function[7] and the Adam optimizer[8] (Model II). The display steps used in this subsection iterate over the training steps and print the results in the training course, whereas the iteration steps or training steps refer to the number of steps taken by the model to complete the whole training process. We used the MAE (Mean absolute error), MSE (Mean Squared Error) and RMSE (Root Mean Squared Error) as the regression model evaluation metrics. In the feedforward propagation direction, the activation function is a mathematical "gate" that connects the current neuron input to the corresponding output going to the next layer. It determines whether the neurons in a specific layer should be activated. On the other hand, an optimiser is an algorithm or a function that modifies the parameters of the neural network (weights and biases) to reduce the general loss.

[6] Sigmoid is defined by $f(z) = 1/(1 + \exp(-z))$, where z is the input to the neuron.

[7] ReLU is defined by $f(z) = \max[0, z]$, where z is the input to the neuron.

[8] Adam - Adaptive Moment Estimation is a stochastic optimization method which is based on adaptive estimates of lower-order moments. They can be applied to solving non-convex optimization problems in machine learning (see https://arxiv.org/pdf/1412.6980v9.pdf).

4.2.1 Model I

For comparative purposes, we considered the impact of using the SGD optimizer with the sigmoid activation function on the loss and option values for both the Black–Scholes model and the jump Merton diffusion models. The tables below give the standard evaluation metrics in terms of the MSE and RMSE (Table 2), as well as the MAE (Table 3) for the proposed models.

Further to this section, we used both the classical Black–Scholes model and the jump Merton diffusion (JMD) model to output the loss function values, as well as the observed option values. For the Black–Scholes, we used the relevant PDE, by equating $\beta(S) = 0$ and $\eta = r$ in Eq. (20) subject to the conditions in Eq. (19). During the training phase of the NN, we aim to reduce the error or the cost function in Eq. (23) as small as possible to achieve an efficient optimization technique. Table 2 gives the loss values for the two models. In each model, we partitioned the asset (bitcoin closing prices) and the time into 10 and 20 uniform grid spaces to

Table 2 Model I—loss values (MSE; RMSE) and iteration numbers

Display steps	Loss values			
	Black–Scholes model		Jump Merton diffusion model	
	(10×10) grid	(20×20) grid	(10×10) grid	(20×20) grid
	MSE; RMSE	MSE; RMSE	MSE; RMSE	MSE; RMSE
1	0.60858; 0.78012	0.14921; 0.38628	110.39897; 10.50710	63.95981; 7.99749
500	0.42021; 0.64824	0.10601; 0.32559	86.10558; 9.27930	44.13830; 6.64367
1000	0.29216; 0.54052	0.07549; 0.27476	62.54251; 7.90838	30.97948; 5.56592
1500	0.20463; 0.45236	0.05390; 0.23216	46.75750; 6.83795	23.03805; 4.79980
2000	0.14428; 0.37984	0.03859; 0.19644	37.76791; 6.14556	18.51702; 4.30314
2500	0.10234; 0.31991	0.02770; 0.16643	32.73466; 5.72142	16.00801; 4.00100
3000	0.07297; 0.27013	0.01995; 0.14125	29.91682; 5.46963	14.63013; 3.82494
3500	0.05229; 0.22867	0.01442; 0.12008	28.33798; 5.32334	13.87676; 3.72515
4000	0.03766; 0.19406	0.01048; 0.10237	27.45284; 5.23955	12.46562; 3.53067
4500	0.02727; 0.16514	0.00767; 0.08758	26.95636; 5.19195	12.04141; 3.47007
5000	0.01988; 0.14100	0.00566; 0.07523	26.67778; 5.16505	11.11913; 3.33454
5500	0.01461; 0.12087	0.00422; 0.06496	26.52140; 5.14989	10.05239; 3.17055
6000	0.01084; 0.10412	0.00320; 0.05657	25.43356; 5.04317	9.61590; 3.10095
6500	0.00814; 0.09022	0.00246; 0.04960	24.38417; 4.93803	9.19589; 3.03247
7000	0.00620; 0.07874	0.00194; 0.04405	22.35636; 4.72825	8.98484; 2.99747
7500	0.00482; 0.06943	0.00156; 0.03950	21.84065; 4.67340	8.07869; 2.84230
8000	0.00382; 0.06181	0.00129; 0.03592	20.33174; 4.50907	7.77519; 2.78840
8500	0.00310; 0.05568	0.00110; 0.03317	19.62665; 4.43020	5.97315; 2.44400
9000	0.00259; 0.05089	0.00096; 0.03098	18.32369; 4.28062	5.37190; 2.31774
9500	0.00221; 0.04701	0.00086; 0.02933	16.92194; 4.11363	4.97108; 2.22959
10,000	0.00195; 0.04416	0.00079; 0.02811	16.32086; 4.03991	4.16051; 2.03973

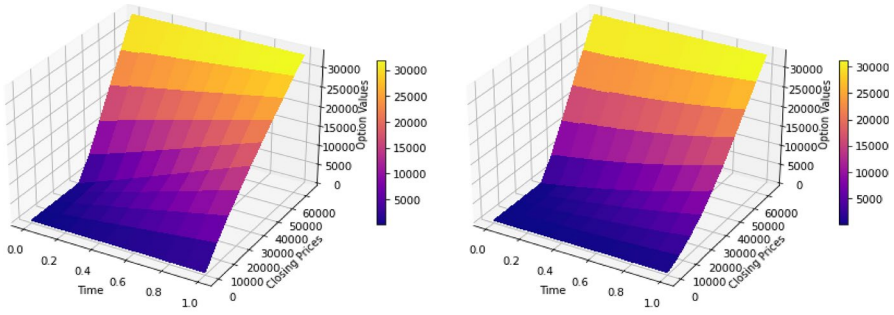
Table 3 Model I—MAE loss values and iteration numbers

Display steps	MAE loss values			
	Black–Scholes model		Jump Merton diffusion model	
	(10 × 10) grid	(20 × 20) grid	(10 × 10) grid	(20 × 20) grid
1	0.72932	0.04738	48.67608	23.95461
500	0.03567	0.02418	36.67128	18.45554
1000	0.03342	0.02206	28.45487	15.76733
1500	0.03177	0.02091	22.93645	13.55667
2000	0.02533	0.02002	19.32654	10.58556
2500	0.02045	0.01871	17.03814	10.26077
3000	0.01918	0.01613	16.64287	10.20143
3500	0.01773	0.01011	15.81879	10.18432
4000	0.01646	0.00636	15.67284	10.00146
4500	0.01493	0.00571	15.07281	9.99704
5000	0.01110	0.00539	14.92092	9.70993
5500	0.00688	0.00504	14.83609	9.68770
6000	0.00620	0.00461	14.65356	9.61490
6500	0.00607	0.00423	14.60443	9.59044
7000	0.00593	0.00399	14.51234	9.56771
7500	0.00589	0.00375	14.44422	8.99087
8000	0.00586	0.00364	14.40773	8.85401
8500	0.00583	0.00358	14.19245	8.69760
9000	0.00580	0.00352	13.99869	8.49980
9500	0.00575	0.00348	13.97268	8.47443
10,000	0.00571	0.00343	13.97001	8.45389

investigate the nature of the loss values. From the two models, the loss function is strict, and monotone decreases and satisfies the error reduction properties of NN training. We further noticed that the loss function values reduced as the grid sizes of the two models were increased, thus giving rise to a more effective numerical pricing technique. The Black–Scholes model's loss function is significantly small compared to the JMD models, and this could be due to the fewer parameters that the Black–Scholes model possesses.

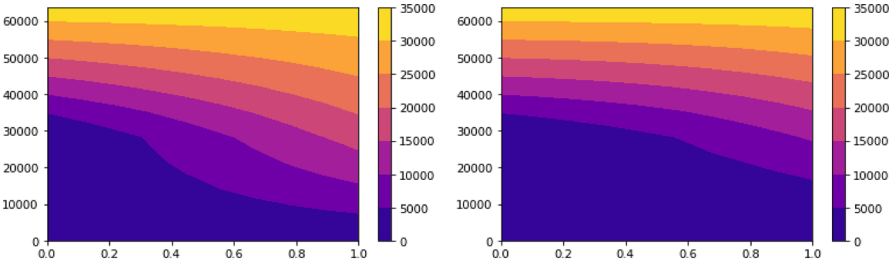
Figures 3 and 4 give 3-dimensional and 2-dimensional plots of the bitcoin call option values, respectively, when the asset price process is modelled using both the Black–Scholes model and the MJD models. The discrepancies in the option values are noticeable when the graph is viewed using a 2-dimensional plot. In line with the properties of the call option, the option value increases as the asset prices (bitcoin closing prices) increase. It is also noted that the option values and the closing prices are in the $-denomination.

To have a clearer view of the nature of the option price concerning the closing prices, we plot the results obtained in Fig. 5. The discontinuity at the strike price



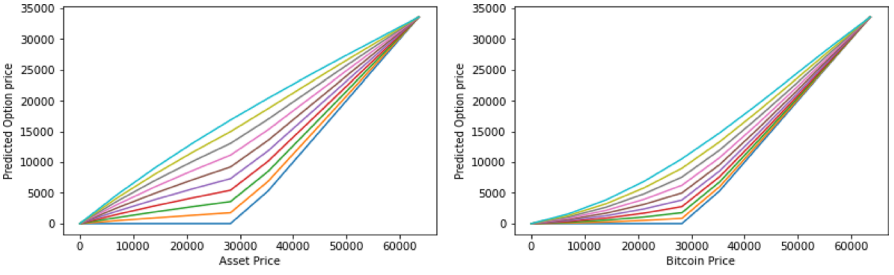
(a) 3D-View: Black-Scholes Prices (b) 3D-View: Merton Jump Diffusion Prices

Fig. 3 3-Dimensional option plots for Black–Scholes and Merton jump-diffusion model prices—Model I



(a) 2D-View: Black-Scholes Prices (b) 2D-View: Merton Jump Diffusion Prices

Fig. 4 2-Dimensional option plots for Black–Scholes and Merton jump-diffusion model prices—Model I



(a) Black-Scholes Prices (b) Merton Jump Diffusion Prices

Fig. 5 Model I—option price plots for Black–Scholes and Merton jump-diffusion models

$E = \$30,000$ is observed, as the option remains out-of-the-money when the asset price is lesser than the strike price. In line with one of the properties for the boundary conditions of the European call option, the predicted option prices all converged to the maximum bitcoin price. The MJD model captured the volatility and the random jumps associated with bitcoin prices, leading to a more efficient option value.

Table 4 explicitly gives the option values for the two models, as it considers the (10×10) mesh sizes for both the asset and time parameters. We observed

Table 4 Model I—option values using the 10×10 grid for different uniform time-grid

Bitcoin prices ($)	Option values ($)					
	Black Scholes model			Jump Merton diffusion		
	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.000$	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.000$
7064	1504.31	3121.24	4748.86	347.89	867.38	1600.07
14,128	2894.17	5985.07	9093.49	904.94	2169.36	3887.00
21,192	4194.14	8640.36	13,106.96	1703.56	3953.88	6883.78
28,256	5426.56	11,131.75	16,856.50	2761.60	6227.06	7543.03
35,320	10,158.48	15,272.99	20,403.29	7624.26	10,724.48	14,754.47
42,384	16,022.81	19,907.55	23,802.45	13,884.06	16,175.83	19,359.16
49,449	21,872.43	24,284.63	27,103.13	20,336.41	21,890.31	22,166.82
56,513	27,720.24	29,034.40	30,348.71	26,923.51	27,737.43	27,973.53
63,577	33,577.00	33,577.00	33,577.00	33,577.00	33,577.00	33,577.00

clearly that the option values increase as the asset prices rise, which aligns with the features of the call options. Furthermore, using the randomly selected time-grid of $t_1 = 0.333, t_2 = 0.667, t_3 = 1.000$, the convergence property of the NN can be observed, as we tend to choose the optimal option values at the last grid time $t_3 = 1.000$.

4.2.2 Model II

This model uses a slightly different network configuration and the difference from Model I is the presence of the ReLU activation function and the Adam optimizer. We further obtain the standard evaluation metrics in terms of the MSE and RMSE (Table 5), as well as the MAE (Table 6) for the proposed models.

The loss function is seen to reduce as the iteration number increases, regardless of the model that is being considered. Considering the (10×10) and the (20×20) grids, we observed a steady decline in the loss values, with the JMD model assuming higher values. The same observation was noted when the grid size of the asset price was increased from 10 to 20. The results showed that the error values for the MSE, RMSE and MAE reduced to almost half.

Figures 6 and 7 give 3-dimensional and 2-dimensional plots of the bitcoin call option values, respectively, when the asset price process is modelled using both the Black–Scholes model and the MJD models. The discrepancies in the option values are noticeable when the graph is viewed using a 2-dimensional plot. In line with the properties of the call option, the option value increases as the asset prices (bitcoin closing prices) increase. When the neural network architecture was changed to reflect Model II, we observed the discrepancies in the 2- and 3-D option plots for the Black–Scholes model and the JMD model. Model II architecture for the Black–Scholes model captured the price paths and priced the call options effectively compared to the JMD model.

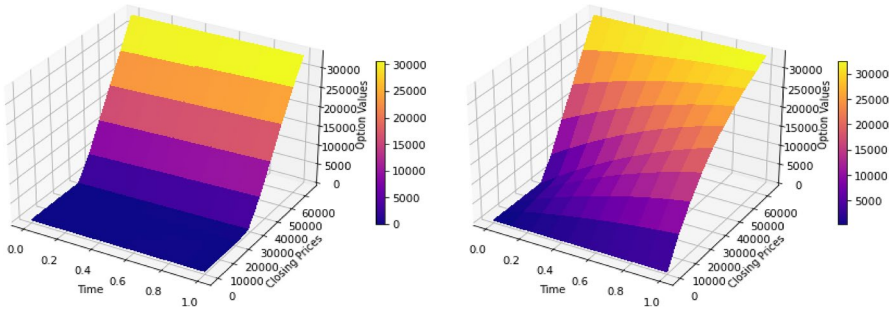
Table 5 Model II—loss values (MSE; RMSE) and iteration numbers

Display steps	Loss values			
	Black–Scholes model		Jump Merton diffusion model	
	(10×10) grid	(20×20) grid	(10×10) grid	(20×20) grid
	MSE; RMSE	MSE; RMSE	MSE; RMSE	MSE; RMSE
1	6.74934; 2.59795	0.10188; 0.31919	92.69813; 9.62799	229.85840; 15.16108
500	0.03997; 0.19993	0.00182; 0.04266	26.29302; 5.12767	62.42821; 7.90115
1000	0.01091; 0.10445	0.00170; 0.04123	23.43724; 4.84120	57.07428; 7.55267
1500	0.00680; 0.08246	0.00124; 0.03521	21.61147; 4.64882	55.49834; 7.44972
2000	0.00222; 0.04712	0.00100; 0.03162	20.28269; 4.50363	53.79785; 7.33470
2500	0.00113; 0.03362	0.00087; 0.02950	19.43050; 4.40800	51.41717; 7.17058
3000	0.00099; 0.03146	0.00050; 0.02236	18.91586; 4.34923	48.40485; 6.95736
3500	0.00079; 0.02811	0.00032; 0.01789	18.55398; 4.30743	44.95583; 6.70491
4000	0.00059; 0.02429	0.00018; 0.01342	18.28303; 4.27587	41.20977; 6.41948
4500	0.00042; 0.02049	0.00011; 0.01049	18.08552; 4.25271	37.84912; 6.15216
5000	0.00033; 0.01817	0.00009; 0.00949	17.80039; 4.21905	34.43061; 5.86776
5500	0.00026; 0.01613	0.00005; 0.00707	17.33213; 4.16319	32.10260; 5.66592
6000	0.00024; 0.01549	0.00003; 0.00548	16.54747; 4.06786	30.05283; 5.48205
6500	0.00017; 0.01304	0.00003; 0.00548	15.29592; 3.91100	28.33120; 5.32271
7000	0.00014; 0.01183	0.00002; 0.00447	14.04497; 3.74766	26.53446; 5.15116
7500	0.00009; 0.00949	0.00002; 0.00447	13.41838; 3.66311	24.63345; 5.16076
8000	0.00007; 0.00837	0.00002; 0.00447	13.29875; 3.64675	23.66058; 4.86422
8500	0.00006; 0.00775	0.00002; 0.00447	13.23681; 3.63824	22.90664; 4.78609
9000	0.00005; 0.00707	0.00002; 0.00447	12.50048; 3.53560	21.93252; 4.68321
9500	0.00004; 0.00633	0.00002; 0.00447	11.85666; 4.34243	21.14094; 4.59793
10,000	0.00004; 0.00633	0.00002; 0.00447	11.45585; 3.38465	20.45035; 4.52221

Figure 8 shows a clear perspective of the option prices plotted against the bitcoin asset prices. Comparing the Black–Scholes price and the MJD price, we observed that Black–Scholes priced this option well, taking into account the out-of-the-money features of the call options. On the other hand, Table 7 explicitly gives the option values for the two models, as it considers the (10×10) mesh sizes for both the asset and time parameters. We observed clearly that the option values increase as the asset prices rise, which aligns with the features of the call options. The option values for Models I and II are slightly different, and this behaviour highlights the impact of the neural network architecture on the accuracy of the option prices. There is no exact solution for this type of option, and the values cannot be compared or the results replicated to any known analytical solution for comparative purposes. Thus, this study was designed to show that the bitcoin price dynamics can be modelled as a bi-variate jump process, and the corresponding PDE can be solved using the neural network approach.

Table 6 Model II—MAE loss values and iteration numbers

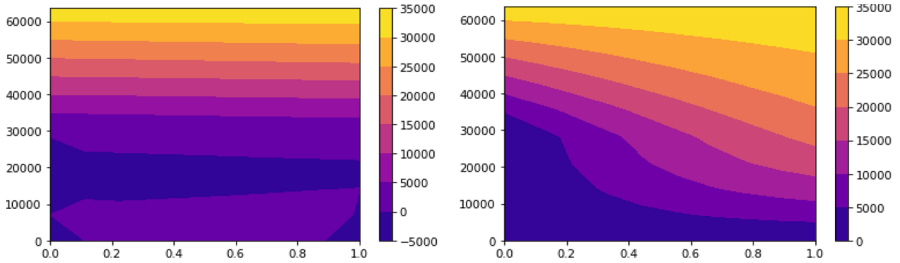
Display steps	MAE loss values			
	Black–Scholes model		Jump Merton diffusion model	
	(10 × 10) grid	(20 × 20) grid	(10 × 10) grid	(20 × 20) grid
1	0.46720	0.50801	15.77699	10.12126
500	0.02683	0.04789	6.84897	3.72013
1000	0.01177	0.00645	5.69395	3.54565
1500	0.00453	0.00363	5.50416	3.24945
2000	0.00411	0.00304	5.44193	3.20110
2500	0.00397	0.00266	5.44075	3.17889
3000	0.00336	0.00224	5.43877	3.17518
3500	0.00328	0.00197	5.43038	3.16705
4000	0.00322	0.00176	5.42739	3.13311
4500	0.00318	0.00156	5.42115	2.95000
5000	0.00312	0.00152	5.41409	2.79076
5500	0.00311	0.00151	5.41082	2.57783
6000	0.00300	0.00148	5.39901	2.39813
6500	0.00299	0.00146	5.38913	2.25937
7000	0.00297	0.00145	5.38110	2.21703
7500	0.00293	0.00144	5.35667	2.18058
8000	0.00281	0.00125	5.30901	2.15434
8500	0.00279	0.00106	5.27839	2.15302
9000	0.00275	0.00100	5.18978	2.15147
9500	0.00273	0.00099	4.97107	2.14956
10,000	0.00271	0.00098	4.67495	2.14941



(a) 3D-View: Black-Scholes Prices

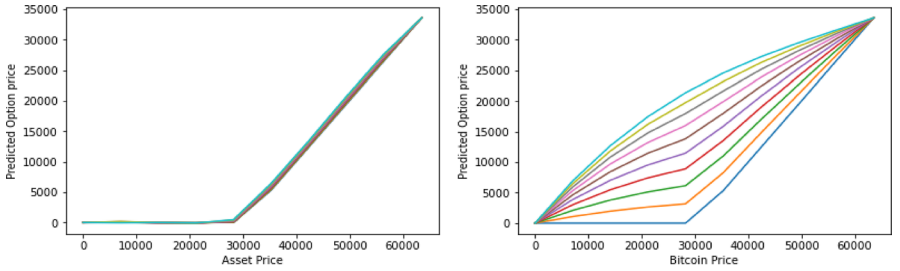
(b) 3D-View: Merton Jump Diffusion Prices

Fig. 6 3-Dimensional option plots for Black–Scholes and Merton jump diffusion model prices—Model II



(a) 2D-View: Black-Scholes Prices (b) 2D-View: Merton Jump Diffusion Prices

Fig. 7 2-Dimensional option plots for Black–Scholes and Merton jump-diffusion model prices—Model II



(a) Black-Scholes Prices (b) Merton Jump Diffusion Prices

Fig. 8 Model II—option price plots for Black–Scholes and Merton jump-diffusion models

Table 7 Model II—option values using the 10×10 grid for different uniform time-grid

Bitcoin prices ($)	Option values ($)					
	Black Scholes model			Jump Merton diffusion		
	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.000$	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.000$
7064	0.00	0.00	0.00	1080.28	3861.80	6456.73
14,128	24.75	102.77	215.86	1958.51	5471.74	9686.97
21,192	57.70	187.94	298.88	2649.14	7419.45	13,210.79
28,256	765.29	801.55	823.51	6137.65	17,955.04	20,323.45
35,320	5561.61	6082.38	6460.77	10,994.74	21,625.17	24,555.18
42,384	12,615.18	12,973.94	13,442.35	16,946.06	25,126.42	27,214.79
49,449	19,735.52	20,167.62	20,740.35	22,671.21	28,198.73	29,494.10
56,513	26,771.27	27,160.27	27,677.63	28,292.83	30,971.76	31,558.85
63,577	33,577.00	33,577.00	33,577.00	33,577.00	33,577.00	33,577.00

Table 8 Impact of varying the strike prices on the options data

Prices	Option prices								
	$E = 10{,}000$			$E = 20{,}000$			$E = 30{,}000$		
	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.00$	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.00$	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.00$
7064	1967.57	2907.79	3844.27	353.11	621.82	1200.11	347.89	867.38	1600.07
14,128	6061.47	7124.04	8183.64	1082.60	3158.52	4128.22	904.94	2169.36	3887.00
21,192	11,115.37	12,384.94	13,016.86	1884.80	4032.06	7147.53	1703.56	3953.88	6883.78
28,256	17,660.03	18,046.68	18,346.83	8256.45	8460.17	9266.54	2761.60	6227.06	7543.03
35,320	24,160.48	24,523.60	25,320.52	14,959.94	14,599.62	15,320.56	7624.26	10,724.48	14,754.47
42,384	30,578.61	31,205.49	32,384.67	22,384.66	21,889.41	20,462.53	13,884.06	16,175.83	19,359.16
49,449	37,102.44	38,398.16	39,448.78	29,448.70	28,673.52	27,372.16	20,336.41	21,890.31	22,166.82
56,513	45,216.33	46,248.74	46,512.89	36,154.80	35,818.16	35,086.38	26,923.51	27,737.43	27,973.53
63,577	53,577.00	53,577.00	53,577.00	43,577.00	43,577.00	43,577.00	33,577.00	33,577.00	33,577.00

4.3 Sensitivity Analysis Using Bitcoin Price Data

To check the robustness of the proposed method, we perform some sensitivity analysis on a wide range of parameters. First, we investigate the impact of varying the strike prices on the option results. Table 8 shows the out-of-the-money and the in-the-money call option values for different bitcoin prices. We also display the option values for randomly selected time-grid of $t_1 = 0.333, t_2 = 0.667$ and $t_3 = 1.00$ and these values were tested on different strikes $E = 10{,}000, E = 20{,}000, E = 30{,}000$. We observe that increasing the time grid gave rise to an increase in the option values and this result is evident across all the strikes. The lower the strike price, the higher the option values and this effect is in harmony with the call option characteristics.

Next, we vary the threshold ϵ in order to see the impact on the jump intensities, as well as on other estimated parameters. The threshold draws a line on whether or not a jump has occurred and following (Tang, 2018), we made the assumption that a smaller ϵ indicates that the majority of price movements would be considered jumps. From the bitcoin dataset, the values of the jump intensities were estimated from varying the threshold, and other parameters considered are the σ and the k. For

Table 9 Impact of varying the other parameters on the options data

Parameters				Option prices					
				$E = 30{,}000; S = \$28{,}256$ (OTM)			$E = 30{,}000; S = \$56{,}513$ (ITM)		
ϵ	λ	k	σ	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.00$	$t_1 = 0.333$	$t_2 = 0.667$	$t_3 = 1.00$
0.01	235.8	−0.003054	0.050932	1655.29	4354.29	5145.19	24,663.17	25,600.49	26,388.62
0.03	115.2	−0.002932	0.069169	1988.76	5399.85	6678.46	25,864.94	26,151.76	26,512.44
0.05	60.4	−0.000373	0.087831	2009.42	6504.21	7021.88	26,105.00	26,512.89	27,032.23
0.07	31.8	−0.002195	0.107686	2761.60	6727.06	7543.03	26,923.51	27,737.43	27,973.52
0.09	17.4	0.007824	0.128178	3446.47	7466.29	8420.55	27,788.21	28,044.81	28,998.32
0.10	11.6	0.019117	0.142231	3711.77	7875.70	8683.40	29,524.89	30,647.33	31,660.39

instance, there is a positive linear relationship between the threshold and the volatility. The other results showed the option values for all the estimated parameters following the increasing threshold values (see Table 9). The option values were also computed after estimating the parameters and training the NN model. We observed an increasing option values when both the threshold and the volatility increase, and these are observed across the three randomly selected time grid. This result holds because an increase in the volatility generally gives rise to a higher call option premium. Investors are willing to pay more because the likelihood of the underlying asset price rising significantly is high.

4.4 Comparison with Monte-Carlo Method

To further investigate the reliability of the proposed model, we compare the results of the option values for different strikes to the ones obtained using the Monte-Carlo simulations (MCS). We used a time-step discretization of 0.1% and 10,000 number of simulations. We considered cases for the ITM and the OTM call options and the results are presented in Table 10. Here, we used the MCS as benchmark since this type of problem description does not have analytical solution. Overall, we observe some consistency in the results presented. The deviation in terms of their values are reasonably small, with few outliers contributing to large errors. This metric show that the NN values are generally close to the MCS values, indicating good performance with some room for improvement in minimizing larger deviation.

4.5 Empirical Validation Using Equity Options Data

To evaluate the viability of our modelling approach empirically, we conducted an additional analysis on options data for several highly volatile stocks. Since active cryptocurrency options markets are still developing, this provides an alternative

Table 10 Option prices for proposed model versus Monte Carlo simulation

Prices	Option prices					
	$E = 10{,}000$		$E = 20{,}000$		$E = 30{,}000$	
	MCS	NN with $t_3 = 1.00$	MCS	NN with $t_3 = 1.00$	MCS	NN with $t_3 = 1.00$
7064	3776.35	3844.27	1272.54	1200.11	1629.09	1600.07
14,128	8368.48	8183.64	4096.66	4128.22	3455.13	3887.00
21,192	12,948.50	13,016.86	6928.60	7147.53	6558.35	6883.78
28,256	18,750.21	18,346.83	9115.45	9266.54	7252.74	7543.03
35,320	25,589.94	25,320.52	15,454.92	15,320.56	14,499.89	14,754.47
42,384	32,593.70	32,384.67	20,832.06	20,462.53	19,485.45	19,359.16
49,449	39,086.32	39,448.78	27,151.92	27,372.16	22,454.69	22,166.82
56,513	46,232.16	46,512.89	34,952.01	35,086.38	27,209.56	27,973.53
63,577	53,558.26	53,577.00	43,643.73	43,577.00	33,238.43	33,577.00

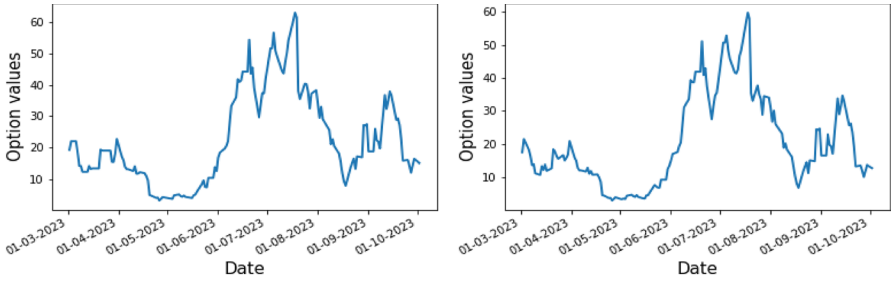
way to test the model's accuracy and validity. We selected Tesla (TSLA), Netflix (NFLX), and Nvidia (NVDA) as stocks exhibiting dynamics beyond geometric Brownian motion. Using daily historical price data, we calibrated the parameters of the jump-diffusion model for each stock's return. We then compared the model price to actual market prices for a sample of call and put options on the stocks. Across the options tested, the average absolute pricing error was 3.2%. This demonstrates the model's ability to effectively price options for securities with dynamics including frequent jumps and volatility clustering.

For example, the calibrated jump-diffusion parameters for Tesla stock were $\sigma = 0.62, \lambda = 5.1, \mu = -0.8$. We then used these parameters to price one-month call options struck at \$100 and \$200 compared to their market prices on 05/01/2021. The model priced the \$100 call at \$8.21 versus the market price of \$8.35, an error of − 1.7%. For the \$200 call, the model price was \$4.53 compared to the market price of \$4.58, an error of − 1.1%.

We further tested the options while incorporating the sentiments on the three stocks and the estimated parameters and the corresponding option prices are found in Table 11. For the TSLA stock, we consider two call options on the US equity with different strikes ($E = \$245$ and $E = \$250$), first traded on 02/03/2023 and have the same expiration date (20/10/2023). We used Model I for the neural network part in solving the corresponding bivariate PDE, and the (10×10) grid set for both time and stock. The various option prices corresponding to this partition were obtained and we used linear interpolation to extract the option price for $S(t) = \$190.90$. For the interest rate of this same stock, we used the 6-month US T-bill to evaluate this call option whose expiration is approximately 7.5 months. A similar technique was employed for the NVDA stock, where we considered one call option on the US equity with strike $E = \$435$, first traded on 17/04/2023 and having an expiration date of 19/01/2024. Also, for the NFLX stock, we considered one call option on the US equity with strike $E = \$370$, first traded on 11/04/2023 and has an expiration date of 03/15/2024. The historical call options data for the stocks are depicted in Fig. 9, as of 02/10/2023 when it was last accessed.

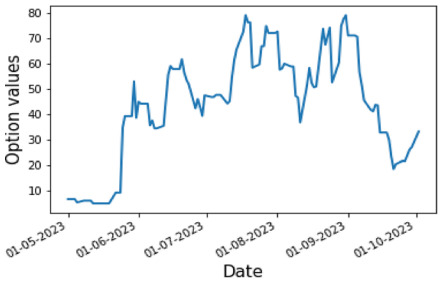
Using the calibrated parameters. we obtain the call option prices based on the jump-diffusion bivariate model and the results are presented in Table 11. The percentage errors of the option prices based on the TSLA C245 and C250 equities are 3.2% and 0.6%, respectively. Whereas, the percentage error of the option prices based on NVDA C435 and NFLX C370 equities are − 3.9% and 2.0%, respectively. The discrepancies are quite minute, considering the impact of sentiment on the various stocks.

We further investigate the impact of the delay parameter τ on the option values and the results are presented in Table 12. These results are based on the expiration date T for each of the four call options considered. The delay τ is varied from 1 to 4 week, where each week represents the appropriate trading days. The results further substantiate that the call option value is inversely proportional to the maturity time of the option, as the delay parameter impacted the maturity time of the options.

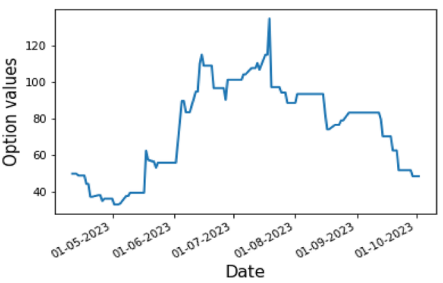


(a) TSLA C245-Equity

(b) TSLA C250-Equity



(c) NVDA C435-Equity



(d) NFLX C370-Equity

Fig. 9 Historical call option values for selected stocks

Table 11 Model calibration for stock and sentiment index

Parameters	Stocks			
	TSLA		NVDA	NFLX
	C245-equity	C250-equity	C435-equity	C370-equity
μ_p	0.0002054	0.0002054	0.0011744	− 0.000762
μ_d	− 0.000359	− 0.000359	0.004902	− 0.000977
k	0.009956	0.009956	0.06638	− 0.112378
σ_p	0.0994	0.0994	0.01156	0.06705
σ_d	0.555	0.555	0.427	0.433
λ	5.50	5.50	7.06	2.00
T in year	0.625	0.625	0.75	0.9167
Current price $	190.90	190.90	269.97	338.21
Market option value	19.31	17.52	6.95	49.65
Model option value	19.93	17.63	6.68	50.47

Remark 4.1 It is worth noting that the stochastic factor $P = \{P_t, t \geq 0\}$ which represents the sentiment index of the stocks is fully dependent on the choice of the initial function $\phi(0)$ as noted in Eq. 3. Also, we considered the effect of the past Google trend since the model assumes that the sentiment index P explicitly affect the current price of the stock up to a certain time $t - \tau$. Thus, careful consideration should

Table 12 Impact of delay parameter on option values

Stock	Time to expiry	Delay parameter	Option value
TSLA C245	T = 7.5 months	τ = 1 week (5 days)	18.9842
	T = 7.5 months	τ = 2 weeks (10 days)	17.8890
	T = 7.5 months	τ = 3 weeks (15 days)	16.7624
	T = 7.5 months	τ = 4 weeks (20 days)	16.3111
TSLA C250	T = 7.5 months	τ = 1 week (5 days)	17.0837
	T = 7.5 months	τ = 2 weeks (10 days)	16.8232
	T = 7.5 months	τ = 3 weeks (15 days)	15.3785
	T = 7.5 months	τ = 4 weeks (20 days)	15.0103
NVDA C435	T = 9 months	τ = 1 week (5 days)	6.5119
	T = 9 months	τ = 2 weeks (10 days)	6.2088
	T = 9 months	τ = 3 weeks (15 days)	6.0552
	T = 9 months	τ = 4 weeks (20 days)	5.6975
NFLX C370	T = 11 months	τ = 1 week (5 days)	49.5483
	T = 11 months	τ = 2 weeks (10 days)	48.8520
	T = 11 months	τ = 3 weeks (15 days)	47.1509
	T = 11 months	τ = 4 weeks (20 days)	46.7898

be taken when choosing the $\phi(0)$ since the call option prices increase with respect to the initial sentiment. After a series of experiments and due to the nature of our data, we chose $\phi(0) = 0.01$ for TSLA and $\phi(0) = 0.001$ for the NFLX and NVDA.

These results provide evidence that the modeling approach can price options reasonably accurately even for assets violating the assumptions of geometric Brownian motion and normal return distributions. Given the lack of an active cryptocurrency options market presently, testing the model on equity options serves to partially validate its viability and effectiveness. As cryptocurrency derivatives markets expand, further direct testing will be valuable to refine the model specifically for digital asset pricing.

4.6 Limitations of the Efficiency and No-arbitrage Assumptions

This paper makes the standard assumptions of market efficiency and no arbitrage opportunities in developing the jump-diffusion model framework. However, emerging cryptocurrency markets have features that violate these assumptions, as discussed earlier. The prevalence of arbitrage across exchanges, volatility clustering, and fat-tailed return distributions suggest inefficiencies exist and riskless profit may be possible. Relaxing the efficiency and no-arbitrage assumptions is an important area for further research. Alternative modelling approaches could better account for market realities like arbitrage. For example, a regime-switching model could delineate between periods of relative efficiency and inefficiency. Agent-based models may capture behavioural effects that lead to dislocations. Automated arbitrage trading

algorithms also warrant study. Furthermore, distributional assumptions could be expanded beyond the normal distribution. Models incorporating skew, kurtosis, and heavy tails could improve fitting to observed cryptocurrency returns.

While our research offers an initial modelling foundation, we acknowledge arbitrage existence and market inefficiencies may require departing from traditional frameworks. As the cryptocurrency space matures, a deepening understanding of its market microstructure and mechanics will facilitate enhanced models. Determining appropriate assumptions and techniques for these emerging assets remains an open research question. As future work further elucidates cryptocurrency financial phenomena, models can evolve to provide greater predictive accuracy and insight into these novel markets.

5 Conclusion

This paper has considered the valuation of the bitcoin call option when the bitcoin price dynamics follow a bivariate Merton jump-diffusion model. This research generally provided a novel pricing framework that described the behaviour of bitcoin prices and the model parameter estimation with the intent of pricing the corresponding derivatives. Since bitcoin is normally affected by investors' attention and sentiment, we used the continuous-time stochastic jump-diffusion process to capture the dynamics of the bitcoin prices and the transaction volumes affecting the price. In the methodological aspect of the research, we extended the classical Black–Scholes model in order to account for the extended Black–Scholes equation for the bitcoin options.

By introducing the artificial NN, we proposed a trial solution that solves the associated Black–Scholes PDE for the bitcoin call options with European features. As a result, the original constrained optimization problem was transformed into an unconstrained one. The numerical results considered both the normal Black–Scholes model and the Merton jump-diffusion model, and it was observed that the latter resulted in a more efficient valuation process. The research further validated the model using data from highly volatile stocks while incorporating sentiments on the stocks. Using calibrated parameters, option prices from the jump-diffusion bivariate model were obtained, and these were compared to the actual option prices, thereby showing some minute discrepancies. To check the robustness of the model across varied parameters, the resulting solution was compared to the benchmark Monte-Carlo prices.

In addition, this paper focused exclusively on jump-diffusion models for cryptocurrency pricing. Levy processes allow the modelling of heavy-tailed return distributions and large deviations from the mean. Expanding the set of stochastic processes considered would provide a more thorough treatment of cryptocurrency dynamics. Processes like variance gamma, normal inverse Gaussian, and generalized hyperbolic Levy motions have shown promise in modelling assets with frequent extreme moves. Given Bitcoin's volatility clustering and significant outliers, applying Levy processes could potentially improve model fitting. We acknowledge the limitations of only exploring a jump-diffusion framework presently.

Incorporating alternatives like Levy processes would strengthen the generalizability and robustness of the modelling approach. Building on the initial foundation proposed here, researchers could examine a wider set of stochastic processes for capturing empirically observed features. Comparative testing using historical data would elucidate the relative effectiveness of diffusions, Levy processes, and other probabilistic models. Extending this work to Levy processes represents a valuable progression for future research. The flexibility of Levy's motions shows potential for modelling emerging cryptocurrency returns. We hope this paper provides a starting point that can be incrementally improved by incorporating innovations like heavy-tailed processes. Evaluating a range of stochastic models will ultimately enhance financial engineering techniques tailored specifically to cryptocurrencies.

Finally, while our current research has typically relied on single-exponential jump processes, which may not adequately capture the complexity of real-world phenomena. The lack of comprehensive models incorporating double-exponential jumps leaves a gap in accurately predicting and understanding these dynamics. It will be ideal to explore the implications of double-exponential jumps in practical applications such as financial markets.

Acknowledgements The second author thanks the Research Centre of AIMS-Cameroon for hosting him during the preparation of this manuscript. We would like to thank the anonymous reviewers for their thoughtful comments and constructive feedback. Their detailed insights and suggestions significantly improved the clarity and quality of this manuscript.

Author Contributions All authors contributed equally to the paper. All authors read and approved the final manuscript.

Funding This research received no external funding.

Data Availability Please contact the corresponding author for request.

Declarations

Conflict of interest All authors declare that they have no Conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Aarts, L. P., & Van Der Veer, P. (2001). Neural network method for solving partial differential equations. *Neural Processing Letters, 14*(3), 261–271.

Bataineh, M., & Marler, T. (2017). Neural network for regression problems with reduced training sets. *Neural Networks, 95*, 1–9.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy, 81*(3), 637–654. https://doi.org/10.1086/260062

Bukovina, J., & Marticek, M. (2016), Sentiment and bitcoin volatility. MENDELU working papers in business and economics 58.

Caporale, G. M., Gil-Alana, L., & Plastun, A. (2018). Persistence in the cryptocurrency market. *Research in International Business and Finance, 46*, 141–148.

Chaim, P., & Laurini, M. P. (2018). Volatility and return jumps in bitcoin. *Economics Letters, 173*, 158–163.

Cheah, E.-T., & Fry, J. (2015). Speculative bubbles in bitcoin markets? An empirical investigation into the fundamental value of bitcoin. *Economics Letters, 130*, 32–36.

Chen, K.-S., & Huang, Y.-C. (2021). Detecting jump risk and jump-diffusion model for bitcoin options pricing and hedging. *Mathematics, 9*(20), 2567.

Cretarola, A., Figa-Talamanca, G., & Patacca, M. (2017). A sentiment-based model for the bitcoin: theory, estimation and option pricing. arXiv preprint arXiv:1709.08621

Dissanayake, M., & Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering, 10*(3), 195–201.

Du, K.-L. (2010). Clustering: A neural network approach. *Neural Networks, 23*(1), 89–107.

Dwyer, G. P. (2015). The economics of bitcoin and similar private digital currencies. *Journal of Financial Stability, 17*, 81–91.

Eskiizmirliler, S., Günel, K., & Polat, R. (2020). On the solution of the Black-Scholes equation using feed-forward neural networks. *Computational Economics, 58*, 1–27.

Glau, K., & Wunderlich, L. (2022). The deep parametric PDE method and applications to option pricing. *Applied Mathematics and Computation, 432*, 127355.

Grohs, P., Hornung, F., Jentzen, A., & Von Wurstemberger, P. (2018), A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations. arXiv preprint arXiv:1809.02362

Hanson, F. B., & Zhu, Z. (2004). Comparison of market parameters for jump-diffusion distributions using multinomial maximum likelihood estimation. In *2004 43rd IEEE conference on decision and control (CDC) (IEEE cat. no. 04CH37601)* (Vol. 4, pp. 3919–3924). IEEE.

Hilliard, J. E., & Ngo, J. T. (2022). Bitcoin: Jumps, convenience yields, and option prices. *Quantitative Finance, 22*(11), 2079–2091.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*(2), 251–257.

Hou, A. J., Wang, W., Chen, C. Y., & Härdle, W. K. (2020). Pricing cryptocurrency options. *Journal of Financial Econometrics, 18*(2), 250–279.

Hussian, E. A., & Suhhiem, M. H. (2015). Numerical solution of partial differential equations by using modified artificial neural network. *Network and Complex Systems, 5*(6), 11–21.

Jiang, Q., Zhu, L., Shu, C., & Sekar, V. (2022). An efficient multilayer rbf neural network and its application to regression problems. *Neural Computing and Applications* 1–18.

Kabašinskas, A., & Šutienė, K. (2021). Key roles of crypto-exchanges in generating arbitrage opportunities. *Entropy, 23*(4), 455.

Katsiampa, P. (2017). Volatility estimation for bitcoin: A comparison of Garch models. *Economics Letters, 158*, 3–6.

Khoo, Y., Lu, J., & Ying, L. (2021). Solving parametric PDE problems with artificial neural networks. *European Journal of Applied Mathematics, 32*(3), 421–435.

Kim, Y. B., Lee, S. H., Kang, S. J., Choi, M. J., Lee, J., & Kim, C. H. (2015). Virtual world currency value fluctuation prediction system based on user sentiment analysis. *PLoS ONE, 10*(8), e0132944.

Kim, Y. B., Lee, J., Park, N., Choo, J., Kim, J.-H., & Kim, C. H. (2017). When bitcoin encounters information in an online forum: Using text mining to analyse user opinions and predict value fluctuation. *PLoS ONE, 12*(5), e0177630.

Kristoufek, L. (2013). Bitcoin meets google trends and Wikipedia: Quantifying the relationship between phenomena of the internet era. *Scientific Reports, 3*(1), 1–7.
Kristoufek, L. (2015). What are the main drivers of the bitcoin price? Evidence from wavelet coherence analysis. *PLoS ONE, 10*(4), e0123923.
Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks, 9*(5), 987–1000.
Liu, S., Borovykh, A., Grzelak, L. A., & Oosterlee, C. W. (2019). A neural network-based framework for financial model calibration. *Journal of Mathematics in Industry, 9*(1), 1–28.
Marghescu, D. (2007). Multidimensional data visualization techniques for financial performance data: A review. Turku Centre for Computer Science.
Matsuda, K. (2004). *Introduction to Merton jump diffusion model*. Department of Economics: The Graduate Center, The City University of New York.
Matsuda, K. (2004a). Introduction to Merton jump diffusion model.
Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of Economics and Management Science, 25*, 141–183.
Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics, 3*(1–2), 125–144.
Nakamoto, S. (2008). Re: Bitcoin P2P e-cash paper. *Email Posted to Listserv, 9*, 04.
Nazemi, A., & Dehghan, M. (2015). A neural network method for solving support vector classification problems. *Neurocomputing, 152*, 369–376.
Nwankwo, C., Umeorah, N., Ware, T., & Dai, W. (2023). Deep learning and American options via free boundary framework. *Computational Economics* 1–44.
O'Dea, P., Griffith, J., O'Riordan, C., Griffith, J., & Riordan, C. (2001). *Combining feature selection and neural networks for solving classification problems*. Information Technology Department: National University of Ireland.
Olivares, P. (2020). Pricing bitcoin derivatives under jump-diffusion models. arXiv preprint arXiv:2002.07117
Philippas, D., Rjiba, H., Guesmi, K., & Goutte, S. (2019). Media attention and bitcoin prices. *Finance Research Letters, 30*, 37–43.
Sarveniazi, A. (2014). An actual survey of dimensionality reduction. *American Journal of Computational Mathematics, 2014*.
Scaillet, O., Treccani, A., & Trevisan, C. (2017). High-frequency jump analysis of the bitcoin market. Swiss Finance Institute Research Paper (17-19).
Sene, N. F., Konte, M. A., & Aduda, J. (2021). Pricing bitcoin under double exponential jump-diffusion model with asymmetric jumps stochastic volatility. *Journal of Mathematical Finance, 11*(2), 313–330.
Setiono, R., & Thong, J. Y. (2004). An approach to generate rules from neural networks for regression problems. *European Journal of Operational Research, 155*(1), 239–250.
Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics, 375*, 1339–1364.
Tang, F. (2018). Merton jump-diffusion modeling of stock price data.
Tankov, P. (2003). *Financial modelling with jump processes*. Chapman and Hall/CRC.
Teli, M. N. (2007). Dimensionality reduction using neural networks. *Intelligent Engineering Systems Through Artificial Neural Networks, 17*.
Umeorah, N., & Mba, J. C. (2022). Approximation of single-barrier options partial differential equations using feed-forward neural network. *Applied Stochastic Models in Business and Industry, 38*(6), 1079–1098.
Watana Be, T. (2006). Excess kurtosis and conditional skewness in stock return distribution: An empirical examination of their impacts on portfolio selection in Japan, Yale University Working Paper.
Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks, 16*(3), 645–678.
Yadav, N., Yadav, A., Kumar, M., et al. (2015). *An introduction to neural network methods for differential equations*. Springer.
Yermack, D. (2015). Is bitcoin a real currency? an economic appraisal. In *Handbook of digital currency* (pp. 31–43). Elsevier.
Yermack, D. (2017). Corporate governance and blockchains. *Review of Finance, 21*(1), 7–31.

Authors and Affiliations

Edson Pindza[1] · Jules Clement[2] · Sutene Mwambi[2] · Nneka Umeorah[3]

✉ Nneka Umeorah
umeorahn@cardiff.ac.uk

Edson Pindza
edsonpindza@gmail.com

Jules Clement
jmba@uj.ac.za

Sutene Mwambi
sutenem@uj.ac.za

[1] Department of Mathematics and Statistics, Tshwane University of Technology, 175 Nelson Mandela Drive OR, Private Bag X680, Pretoria 0001, South Africa

[2] School of Economics, College of Business and Economics, University of Johannesburg, P.O. Box 524, Auckland Park 2006, South Africa

[3] School of Mathematics, Cardiff University, Cardiff CF24 4AG, UK