



Paper Type: Original Article

Hybrid Generative Adversarial Network (GAN) Framework for Vanilla Option Pricing: Evidence from the NIFTY 50

Anand Sharma^{1,*} , Danny Seow Wei Jie² 

¹ Department of Computer Science, WorldQuant University, USA; anand.sharma9816@gmail.com

² Master of Business Administration, University of the People, USA; DannySeowWeiJie@my.uopeople.edu

Citation:

Received: 10 April 2025
Revised: 18 July 2025
Accepted: 19 January 2026

Sharma, A., & Seow Wei Jie, D. (2026). Hybrid Generative Adversarial Network (GAN) Framework for Vanilla Option Pricing: Evidence from the NIFTY 50. Transactions on Quantitative Finance and Beyond, 3(3), 239–261

Abstract

This study presents a hybrid modeling framework that integrates Generative Adversarial Networks (GANs) with Monte Carlo simulation to improve the pricing of vanilla options on the NIFTY 50 index. Traditional pricing models often rely on parametric assumptions that may not fully capture the nonlinear dependencies, volatility clustering, and regime shifts characteristic of emerging markets. GANs, commonly known for generating images in AI applications, offer a data-driven alternative by learning latent structures from historical market data and generating synthetic asset paths that reflect realistic statistical properties. These high-fidelity trajectories are then used as inputs to a Monte Carlo pricing engine for standard European-style call options. Empirical results demonstrate that the GAN-based approach yields comparable pricing accuracy to conventional methods, with occasional outperformance in distributional realism — albeit at the cost of significantly higher training time due to its iterative and data-intensive nature. This hybrid methodology bridges deep learning and financial engineering, offering a scalable and adaptive solution for derivative pricing in data-rich environments. The framework also holds promise for broader applications in risk management, portfolio stress testing, and market scenario analysis.

Keywords: Generative adversarial networks, Vanilla options, Black-scholes model, Heston model, Monte Carlo simulation.

1|Introduction

The accuracy of vanilla options pricing is fundamental to modern finance, serving as the foundation for risk management and derivative trading strategies [1], which is especially true for European-style calls and



Corresponding Author: anand.sharma9816@gmail.com



<https://doi.org/10.22105/tqfb.vi.91>



Licensee System Analytics. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

puts. Traditional options pricing models, such as the Black–Scholes–Merton (BSM) framework, operate under simplifying assumptions such as constant volatility and log-normal distribution of asset returns [2]. While elegant, these assumptions fail to capture critical real-world phenomena observed in equity indices such as NIFTY 50 in India, including heavy tails (leptokurtosis), volatility clustering, and persistent volatility smile/skew [3]. This might lead to misrepresenting, which further leads to systematic mispricing, in particular for options far from the money or with long maturities. This paper proposes a hybrid framework that leverages Generative Adversarial Networks (GANs) for realistic scenario generation and Monte Carlo simulation for pricing, applied to the NIFTY 50 index.

To overcome the limitations of closed-form solutions, practitioners rely on Monte Carlo (MC) simulation for pricing or scenario generation [4]. However, the quality of MC pricing is entirely dependent on the fidelity of the underlying stochastic process model. Meanwhile, Generative Adversarial Networks (GANs), a class of deep learning models, offer a non-parametric approach to learning the intrinsic probability distribution of complex data [5]. This study looks into the potential of GANs to capture the nonlinear dependencies and non-Gaussian characteristics of NIFTY 50 time series, resulting in a superior market scenario generation for derivative valuation.

Vanilla options, particularly European-style calls and puts, are foundational instruments in financial markets. Accurate pricing of these derivatives is critical for hedging, trading, and risk management. In emerging markets like India, traditional models such as Black–Scholes often underperform due to their inability to capture empirical features like volatility clustering, heavy tails, and regime shifts [6]. This paper proposes a hybrid framework that leverages GANs for realistic scenario generation and Monte Carlo simulation for pricing, applied to the NIFTY 50 index.

2|Literature Review

2.1|Traditional Option Pricing Models

The foundation of modern derivative valuation is the Black–Scholes–Merton (BSM) model that provides an analytical, closed-form solution assuming that the asset price follows a Geometric Brownian Motion (GBM) with constant volatility and risk-free rate [7, 8]. The failure of this assumption to hold in real markets led to the development of more sophisticated methods.

Monte Carlo (MC) simulation became essential for models without closed-form solutions, such as those involving path-dependent features or complex stochastic processes. However, standard MC still requires specifying the underlying stochastic process, often a GBM or a jump-diffusion process, which remains a main source of model risk.

A more advanced framework, Stochastic Volatility Models, try to address the persistent volatility smile by allowing the volatility parameter to evolve randomly over time [9]. While these models significantly improve pricing accuracy compared to BSM, they introduce computational complexity, via complex multi-parameter calibration and still rely on the accurate specification of the volatility dynamics [10].

2.2|Introduction to Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), which was introduced by Goodfellow et al. (2014), are a class of deep learning models designed to generate synthetic data that closely resembles a target distribution [11, 12, 13]. A GAN consists of two neural networks that are simultaneously trained: a Generator (G) and a Discriminator (D).

The generator maps random noise $z \sim P_z(z)$ to synthetic data $G(z)$, while the discriminator attempts to distinguish between real data $x \sim P_{data}(x)$ and generated data $G(z)$.

The training involves a minimax game governed by the objective function defined as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_{data}} [\log D(x)] + \mathbb{E}_{z \sim P_z} [\log(1 - D(G(z)))] \quad (1)$$

This adversarial setup encourages the generator to produce samples that are indistinguishable from real data, while the discriminator improves its classification accuracy. The equilibrium of this game is reached when $D(x) = 0.5$ for all inputs, indicating perfect confusion between real and synthetic data.

However, standard GANs often suffer from mode collapse, unstable gradients, and poor convergence, especially when applied to financial time series, which exhibit nonstationarity, volatility clustering, and heavy tails. To address these challenges, Wasserstein GANs (WGANs) were proposed [14], replacing the Jensen–Shannon divergence with the Wasserstein-1 distance (also known as Earth Mover’s Distance), which is defined as:

$$\mathbf{W}(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \mathbb{P}_r, \mathbb{P}_g} \mathbb{E}_{x, y \sim \gamma} [\|x - y\|] \quad (2)$$

The WGAN objective functions is modified to:

$$\min_G \max_D \mathbb{E}_{x \sim \mathbb{P}_{data}} [D(x)] - \mathbb{E}_{z \sim \mathbb{P}_z} [D(G(z))] \quad (3)$$

where D represents the set of 1-Lipschitz functions, a constraint enforced via weight clipping. This formulation provides smoother gradients and better convergence, making WGANs particularly suitable for generating realistic financial trajectories. In the context of this study, WGANs are used to simulate synthetic NIFTY 50 paths that preserve empirical properties such as autocorrelation, volatility regimes, and tail behavior. These paths serve as inputs to downstream pricing models for vanilla options, enhancing robustness and realism in derivative valuation.

2.3|Machine Learning and GANs in Quantitative Finance

The application of machine learning, such as deep learning in quantitative finance has grown rapidly, spanning across areas from algorithmic trading to derivative pricing [15]. Within this domain, GANs have demonstrated promise for synthesizing realistic market data (e.g., equities, commodities, and derivatives), allowing for robust back-testing and simulation of extreme events [16]. Some of the research also shown that Time-series Generative Adversarial Networks (TimeGAN) can generate multivariate, sequence-dependent data that preserves both the cross-sectional and temporal characteristics of the original data [17]. This success forms the necessary foundation for the current study, highlighting the feasibility of using GANs to model volatility and correlation structures crucial for pricing complex derivatives.

2.4|The Need for Hybrid Models

The literature establishes that while traditional models are interpretable, they are often statistically limited, and pure machine-learning models, though powerful, can lack economically meaningful constraints. A hybrid approach is therefore desirable: one that preserves financial interpretability while leveraging the distribution-learning strengths of modern generative models [18], [19].

Classical models (Black–Scholes, Merton) assume log-normal returns and constant volatility, assumptions frequently violated in practice. Stochastic volatility models (e.g., Heston) and jump-diffusion models attempt to address these shortcomings but remain parametric and can struggle to capture complex empirical features of returns. Recent advances in machine learning, particularly GANs, have shown promise in generating realistic financial time series [20].

In this work we follow a hybrid, practical strategy: we use a GAN to learn realistic return-path distributions and integrate those synthetic scenarios into a Monte Carlo pricing framework for vanilla options. To maintain financial consistency we apply a small set of economically informed corrections — notably an expanding (cumulative) training window and a drift correction to align generated paths with the risk-neutral drift — rather than embedding an explicit parametric model (like GARCH [21]) into the GAN loss. This preserves interpretability (we still benchmark against Black–Scholes and Heston) while benefiting from the GAN’s flexible distributional learning.

3.1|GAN-Based Scenario Generation Framework

To model realistic return dynamics for vanilla option pricing on the NIFTY 50 index, we employ a Generative Adversarial Network (GAN) trained on historical log-returns using an expanding window approach. Each yearly model is trained on all data available up to (but excluding) that year, ensuring the generation process reflects only past information—consistent with a forward-looking pricing setup.

3.2|Model Architecture

The GAN consists of two neural networks [22]:

The generator maps a latent noise vector sampled from a standard normal distribution to a synthetic sequence of log-returns. The architecture consists of a fully connected layer with 64 neurons and ReLU activation, followed by an output layer with T neurons (no activation), where T varies by year between 83 and 87 trading days. The output is reshaped to match the temporal structure of real sequences. Mathematically, the generator function is defined as

$$G(z; \theta_G) = W_3 \text{ ReLU}(W_2 z + b_2) + b_3 \quad (4)$$

where there are 32 latent dimensions, and the learnable parameters are represented by weights and biases.

The discriminator (or critic) evaluates input sequences and outputs a scalar score rather than a probability. The architecture employs 1D convolution with 32 filters (kernel size 3, same padding, ReLU activation) to capture temporal dependencies, followed by flattening and two fully connected layers (32 neurons with ReLU, then a single output neuron with no activation). The critic function is expressed as

$$D(x; \theta_D) = W_5 \text{ ReLU}(W_4 \text{ Flatten}(\text{Conv1D}(x)) + b_4) + b_5 \quad (5)$$

where all learnable parameters include convolutional filters, weights, and biases.

The WGAN training objective is formulated as previously described:

$$\min_G \max_D \mathbb{E}_{x \sim P_{\text{data}}} [D(x)] - \mathbb{E}_{z \sim P_z} [D(G(z))] \quad (6)$$

where the real data distribution, latent noise distribution (standard normal), and the set of 1-Lipschitz functions are represented respectively. This is equivalent to minimizing the Wasserstein-1 distance:

$$W_1(P_{\text{data}}, P_G) = \sup_{D \in \mathcal{L}_1} \mathbb{E}_{x \sim P_{\text{data}}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] \quad (7)$$

The discriminator loss is defined as

$$L_D = \mathbb{E}_{z \sim P_z} [D(G(z))] - \mathbb{E}_{x \sim P_{\text{data}}} [D(x)] \quad (8)$$

which the discriminator aims to minimize, while the generator loss is:

$$L_G = -\mathbb{E}_{z \sim P_z} [D(G(z))] \quad (9)$$

which the generator minimizes to fool the critic into assigning high scores to synthetic data.

For the WGAN objective to approximate the Wasserstein distance, the discriminator must lie within the space of 1-Lipschitz functions. In our implementation, this constraint is enforced via weight clipping: after each discriminator update, all parameters are clipped to the interval $[-0.01, 0.01]$:

$$\theta_D \leftarrow \text{clip}(\theta_D, -0.01, 0.01) \quad (10)$$

This ensures that the gradient of the discriminator with respect to its input remains bounded.

Following the WGAN protocol, the discriminator is updated twice per generator update to maintain a strong critic throughout training. For each batch, we first perform two discriminator updates by sampling real sequences and generated sequences, computing the discriminator loss, updating parameters via gradient descent with Adam optimizer (learning rate 0.0001), and clipping weights. We then perform one generator update by sampling new noise, computing the generator loss, and updating parameters with the same optimizer.

While our implementation uses weight clipping for computational simplicity, we acknowledge that gradient penalty (WGAN-GP) is generally preferred in the literature [23]. Gradient penalty offers smoother gradients, better convergence, and improved sample quality, particularly for high-dimensional distributions. However, it introduces additional computational overhead due to second-order derivatives. For our application—generating relatively short sequences ($T \approx 80\text{--}90$) on CPU—weight clipping provided adequate training stability.

Both networks are optimized using the Adam optimizer. The discriminator is updated twice per generator step to ensure a strong critic during adversarial training.

3.3|Training Methodology

The training follows a cumulative (expanding window) structure:

$$\text{Training Data for Year } t = \{r_\tau \mid \tau < t\} \quad (11)$$

where r_τ are historical log-returns. The data are normalized before training to ensure numerical stability. Each GAN is trained for 400 epochs per year with a batch size of 32. Further details on model hyperparameters are outlined in the reproducibility appendix at the end of this document.

3.4|Drift Correction and Risk-Neutral Adjustment

A fundamental distinction in derivative pricing is the separation between the real-world probability measure ($\mathbb{P}$) and the risk-neutral probability measure ($\mathbb{Q}$) [24]. The GAN, trained on historical NIFTY 50 log-returns, inherently learns the empirical distribution under $\mathbb{P}$, which reflects the actual return dynamics observed in the market. However, option pricing theory requires valuation under $\mathbb{Q}$, where all assets grow at the risk-free rate when discounted appropriately.

Under the risk-neutral measure, the discounted asset price must be a martingale. For a non-dividend-paying asset with spot price S_0 , risk-free rate r , and maturity T , this condition requires:

$$\mathbb{E}^{\mathbb{Q}}[S_T] = S_0 e^{rT}. \quad (12)$$

Equivalently, for log-returns, if we denote the cumulative log-return over the period as the sum of individual returns, the risk-neutral expectation must satisfy:

$$\mathbb{E}^{\mathbb{Q}} \left[\sum_{i=1}^T \tilde{r}_i \right] = rT. \quad (13)$$

This ensures that the expected cumulative growth of the asset under $\mathbb{Q}$ equals the risk-free rate compounded over the time horizon, eliminating any risk premium present in the real-world measure.

The synthetic return paths $\{\tilde{r}_i\}_{i=1}^T$ generated by the GAN reflect $\mathbb{P}$ -dynamics and will generally exhibit a cumulative mean μ_{GAN} that differs from the risk-neutral drift rT . To transform these returns into the $\mathbb{Q}$ -measure, we apply a deterministic drift correction.

After training, 10,000 synthetic return paths are generated per year. For each year t with maturity T (in years) and risk-free rate r , we first compute the mean cumulative log-return across all generated paths:

$$\mu_{\text{GAN}} = \mathbb{E} \left[\sum_{i=1}^T \tilde{r}_i \right], \quad (14)$$

where the expectation is taken over the $N = 10,000$ simulated paths.

To enforce the martingale condition $\sum_{i=1}^T \tilde{r}_i = rT$, we apply a uniform adjustment to each time step. The drift-corrected returns are obtained by

$$\sum_{i=1}^T \tilde{r}_i = rT \quad \Rightarrow \quad \tilde{r}_i = r_i + \frac{rT - \mu_{\text{GAN}}}{T}. \quad (15)$$

This correction distributes the difference between the target risk-neutral drift (rT) and the empirically observed GAN drift (μ_{GAN}) uniformly across all T time steps. Each individual return is shifted by the same constant $(rT - \mu_{\text{GAN}})/T$, ensuring that the cumulative return of every path sums to exactly rT .

This correction method satisfies several desirable properties:

- **Martingale consistency:** By construction, the adjusted cumulative returns satisfy $\sum_{i=1}^T \tilde{r}_i = rT$ for all paths, ensuring the martingale condition is respected.

Distribution preservation: The correction applies a uniform additive shift to all returns, preserving the relative structure of the GAN-learned distribution. Higher-order moments—including volatility, skewness, and kurtosis—remain governed by the generative model’s learned dynamics, while only the location (mean) is adjusted.

Path-wise enforcement: Unlike methods that only adjust the ensemble mean, our approach ensures each individual synthetic path exhibits the correct risk-neutral drift, providing consistency at both the path and distribution levels.

Computational efficiency: The correction is applied post-generation as a simple additive constant, requiring no retraining of the GAN.

Alternative approaches, such as directly training the GAN on risk-neutral returns or embedding the drift constraint into the loss function, were considered but would compromise the GAN’s ability to learn the rich empirical features of the return distribution; particularly volatility clustering and tail behavior. Our approach maintains the flexibility of the generative model while ensuring theoretical consistency with no-arbitrage pricing.

3.5|GAN Architecture Diagram

This diagram below illustrates the architecture and training flow of the WGAN used to generate synthetic log-returns. The generator transforms latent noise into time-series outputs via dense and reshape layers. The discriminator evaluates both real and generated returns using convolutional and dense layers to produce a critic score. The training loop computes losses, updates both networks, and applies weight clipping to enforce Lipschitz continuity. Arrows indicate the adversarial feedback between components, forming a closed optimization cycle.

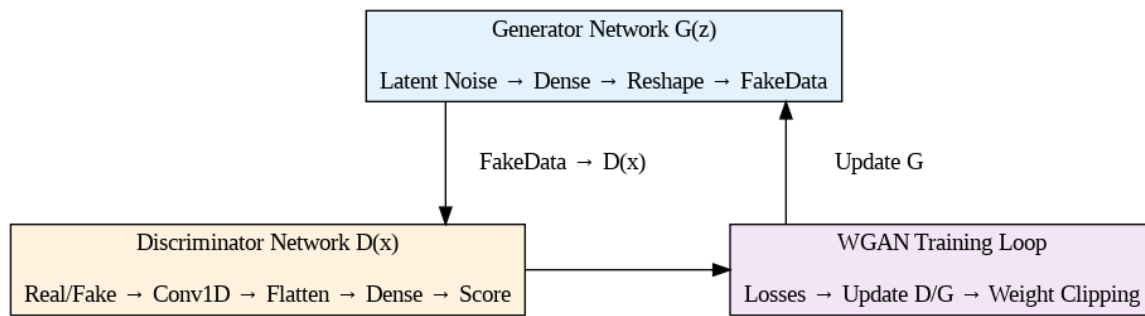


FIGURE 1. GAN architecture diagram (Source: Authors' Own).

3.6|Synthetic Path Generation and Option Pricing

The drift-corrected returns are then used to construct synthetic price paths for Monte Carlo option pricing, as described below:

$$S_t = S_0 \exp \left(\sum_{i=1}^t \tilde{r}_i \right) \quad (16)$$

and fed into a Monte Carlo (MC) pricing module for vanilla European options. This hybrid GAN-MC framework allows the model to generate realistic, non-Gaussian market scenarios while maintaining consistency with financial pricing constraints. Finally, the expected value under the risk-neutral measure ($\mathbb{Q}$) of the discounted payoffs is used for pricing, as described below:

$$C = e^{-rT} \mathbb{E}_{\mathbb{Q}} [\max(S_T - K, 0)] \quad (17)$$

3.7|Benchmark Models

To benchmark the performance of the proposed GAN-based scenario generation framework, we compare it against two classical option pricing models widely used in financial engineering: the Black–Scholes model and the Heston stochastic volatility model.

The Black–Scholes (BS) model assumes that the underlying asset price follows a geometric Brownian motion with constant drift and volatility [25]:

$$dS_t = rS_t dt + \sigma S_t dW_t \quad (18)$$

where r is the continuously compounded risk-free rate, σ is the volatility, and W_t is a standard Wiener process.

The analytical solution for the European call option price is:

$$C = S_0 N(d_1) - Ke^{-rT} N(d_2) \quad (19)$$

$$d_1 = \frac{\ln(S_0/K) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T} \quad (20)$$

where $N(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution.

In our implementation, the volatility parameter σ is both estimated historically (theoretical prices) and inferred from the market—that is, the implied volatility that equates the BS theoretical price to the observed market option price (calibrated prices).

The Heston model extends the BS framework by introducing stochastic volatility. The asset price and variance evolve under correlated stochastic processes:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{v_t} S_t dW_t^S \\ dv_t &= \kappa(\theta - v_t) dt + \sigma_v \sqrt{v_t} dW_t^v \end{aligned} \quad (21)$$

where v_t is the instantaneous variance, κ is the mean-reversion rate, θ is the long-run mean of variance, σ_v is the volatility of volatility, and W_t^S and W_t^v are correlated Wiener processes with correlation coefficient ρ .

Unlike BS, the Heston model captures volatility clustering, leverage effects, and skewness in the return distribution—phenomena commonly observed in equity markets.

Model calibration involves minimizing the error between theoretical option prices (obtained via the semi-analytical Heston pricing formula) and observed market prices across multiple strikes and maturities, yielding an optimal parameter set:

$$\mathbf{f}(\kappa, \theta, \sigma_v, \rho, \nu_0) \mathbf{g} \quad (22)$$

Once calibrated, simulated Heston paths are generated and used to compute model-based distributions of returns, forming a benchmark for comparison against GAN-generated distributions. The distribution process is replicated for theoretical and calibrated BS models, though pricing is carried out via analytical solutions.

3.8|Optimization Method

The Heston model parameters were calibrated to market option prices using a two-stage optimization procedure. First, a global search was performed using differential evolution [26] with 40 iterations and population size 10 to explore the parameter space and avoid local minima. The algorithm minimized the mean squared error between model-predicted and market-observed option prices, subject to bounds ensuring positivity constraints and correlation bounds. Soft penalties were applied for violations of the Feller condition to maintain numerical stability. The best solution from differential evolution was then refined using local L-BFGS-B optimization [27] with a maximum of 500 iterations to achieve precise convergence. This hybrid approach balances global exploration with local refinement.

4|Experiments and Results

4.1|Data Description

This study utilizes two primary datasets:

(1) **NIFTY 50 Index (2015–2025)**

Daily closing prices were obtained from Yahoo Finance (ticker: ^NSEI) [28] and used to compute log returns for training the Generative Adversarial Network (GAN). The same dataset provides the underlying spot price S_0 required for option pricing. Returns are normalized and structured into rolling sequences corresponding to the empirical trading lengths for each year (ranging from 83 to 87 days).

(2) **NSE Option Market Data**

Option prices for the NIFTY 50 were extracted from the National Stock Exchange (NSE) [29]. NIFTY 50 index options traded on the NSE are European-style contracts, eliminating concerns about early exercise that would complicate pricing under standard frameworks. This characteristic ensures compatibility with the Black–Scholes, Heston, and Monte Carlo pricing approaches employed in this study.

Option Selection Criteria

For consistency across years, seven options were selected annually based on the following criteria:

Time to Maturity: Options with approximately three-month time to maturity were selected at the start of each year, targeting contracts expiring near the end of March. Due to trading calendar constraints and data availability (particularly on weekends and holidays), the exact time to maturity varies between 83 and 87 days across different years. This variation reflects the nearest available expiration dates that approximate the target three-month horizon.

Moneyness: Options were selected to span a symmetric range of moneyness levels around the at-the-money (ATM) strike. Moneyness is defined as:

$$\text{Moneyness} = \frac{S_0 - K}{S_0} \quad (23)$$

where K is the strike price and S_0 is the spot price at the time of selection. The chosen options span moneyness levels of -15% , -10% , -5% , 0% , $+5\%$, $+10\%$, and $+15\%$, ensuring both in-the-money (ITM) and out-of-the-money (OTM) contracts are represented. Due to discrete strike price availability on the NSE, the achieved moneyness levels are approximate.

Data Cleaning and Validation

Outlier Removal: Option prices were screened for obvious pricing anomalies. Specifically, contracts that grossly violated the monotonicity principle (i.e., call option prices should generally increase with decreasing strike prices) were flagged and selectively excluded.

Price Verification: The underlying spot prices (S_0) reported in the option data were cross-validated against Yahoo Finance historical prices; no material discrepancies were found.

Bid-Ask Spread: The NSE directly provides closing prices which represent the last traded price or settlement price. As such, bid-ask spread considerations are embedded in the reported closing prices, and no further adjustments were necessary.

Risk-Free Rate

The risk-free rate used in all pricing and drift-correction computations corresponds to the Indian 3-Month Treasury Bill Yield [30]. This rate serves as a proxy for the continuously compounded short-term risk-free rate r in all option pricing models and is assumed constant over each annual calibration window. The rate is specific to each year and reflects the prevailing short-term government borrowing cost.

4.2|Evaluation Metrics

Model performance was assessed across three complementary dimensions:

(1) Pricing Accuracy

Mean Squared Error (MSE) between model-derived and market-observed option prices serves as the primary metric for evaluating pricing performance [31]:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (P_i^{\text{model}} - P_i^{\text{market}})^2 \quad (24)$$

where P_i^{model} and P_i^{market} are theoretical and actual option prices, respectively.

(2) Distributional Fidelity

Beyond price-level accuracy, the statistical similarity between model-simulated and empirically realized return distributions was analyzed. For each year, the first four moments—mean, standard deviation, skewness, and kurtosis—were computed for the synthetic distributions generated by each model (Black–Scholes, Heston, GAN) and compared against market distributions. The total absolute deviation across these moments provides a *Match Score*, where lower values indicate closer alignment with the true data-generating process.

(3) Computational Efficiency

Training and calibration times were recorded to assess the practical feasibility of each model:

Black–Scholes: Executed instantaneously due to its closed-form nature.

Heston Model: Requires numerical calibration of five parameters via iterative optimization (e.g., L-BFGS-B), resulting in moderate computation time.

GAN: Involves deep neural network training across multiple epochs, leading to significantly higher computational cost.

4.3|Key Findings & Discussion

The comparative analysis of option pricing models—Black–Scholes (BS) Theoretical, BS Calibrated, Heston Calibrated, and the Hybrid GAN-Monte Carlo (MC) Framework—yielded several critical observations regarding model performance, computational efficiency, and adherence to market distributions. Visually, all four modeling approaches showed a rough fit to the market-based option prices across the range of strikes. However, quantifiable performance using Mean Squared Error (MSE) revealed significant differences (see Figure 2, Figure 3, Figure 4, and Figure 5).

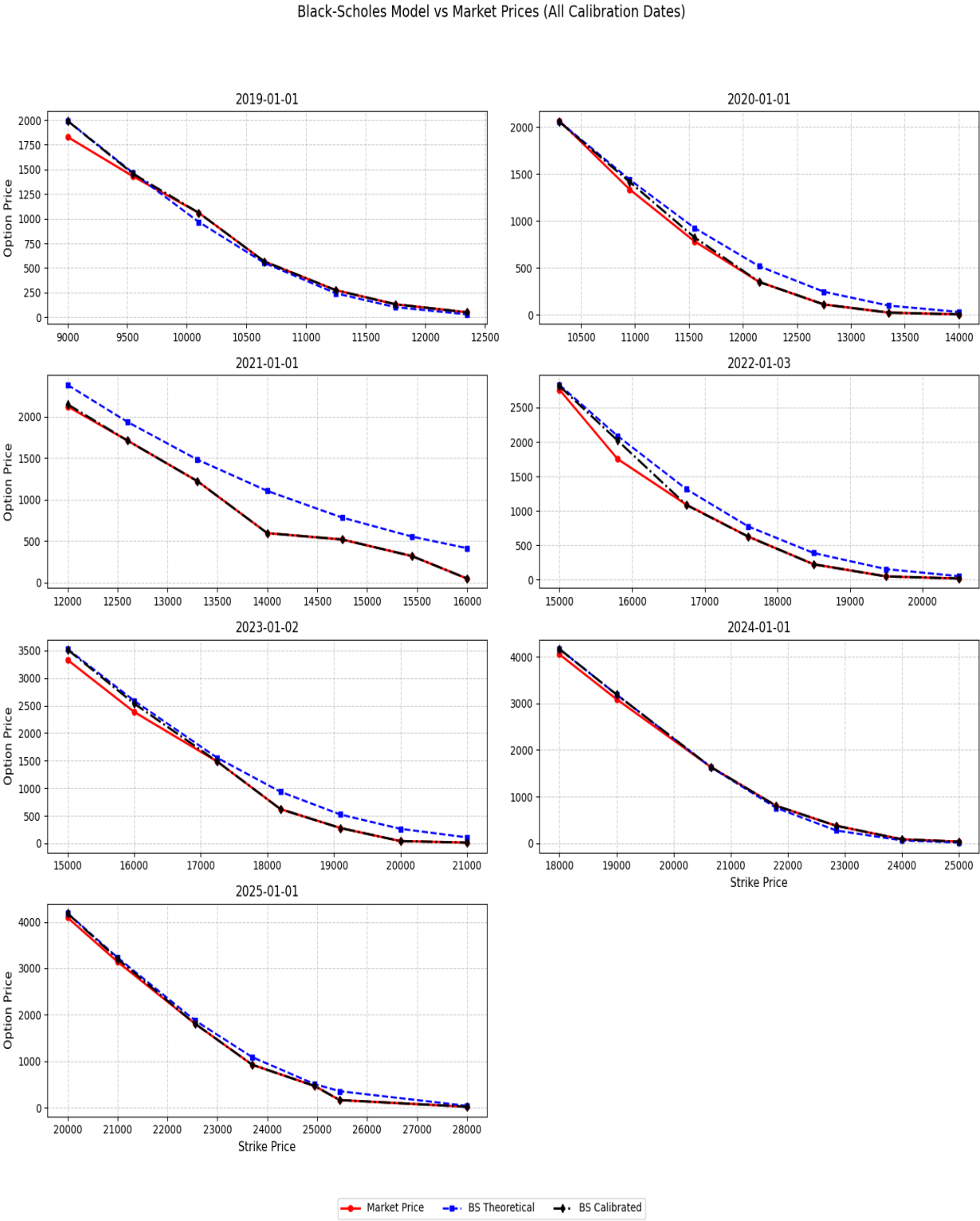


FIGURE 2. Black-Scholes Theoretical & Calibrated Prices vs Market Prices (Source: Authors' Own).

Heston Model vs Market Prices (All Calibration Dates)

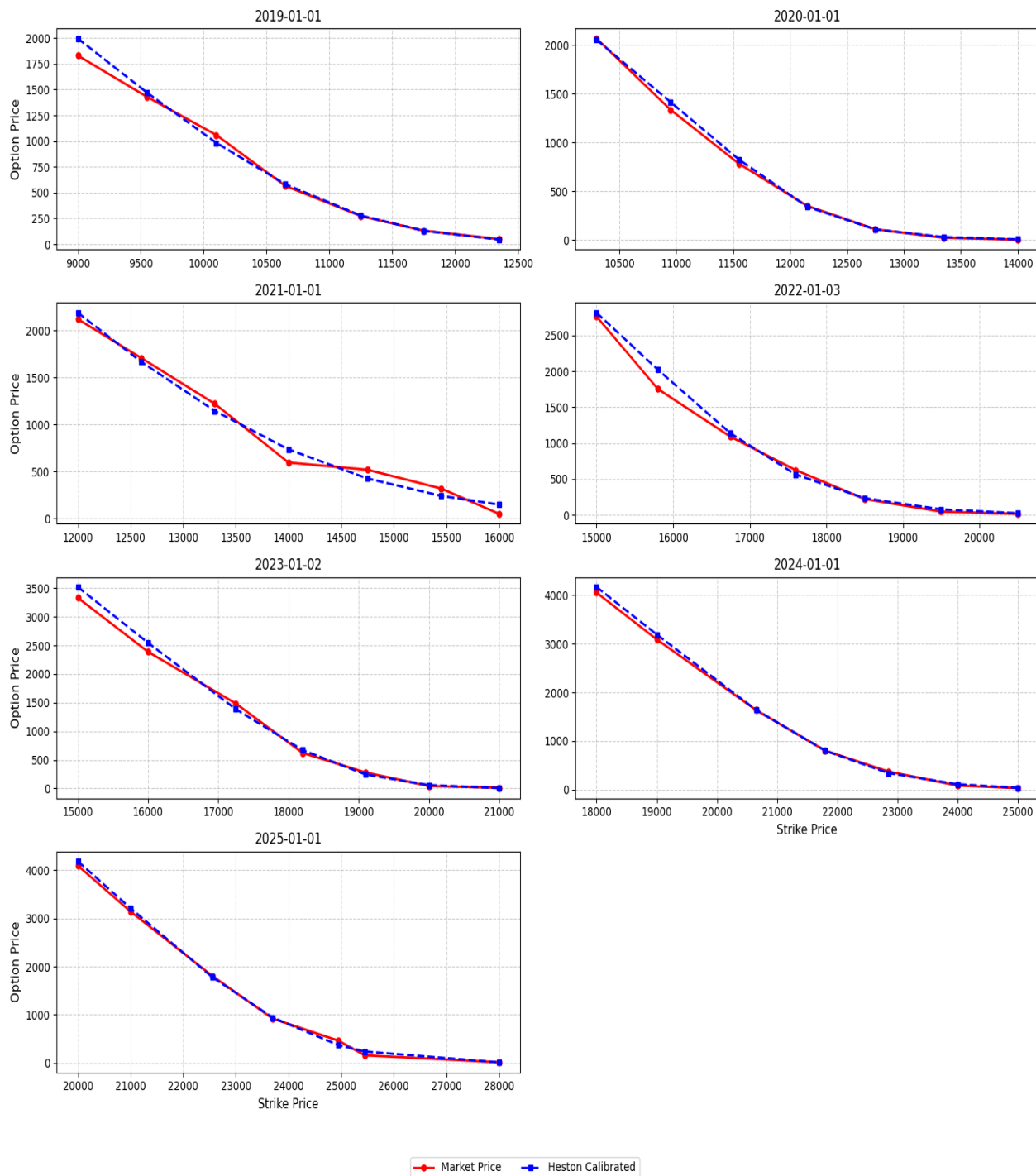


FIGURE 3. Heston Calibrated Prices vs Market Prices (Source: Authors' Own).

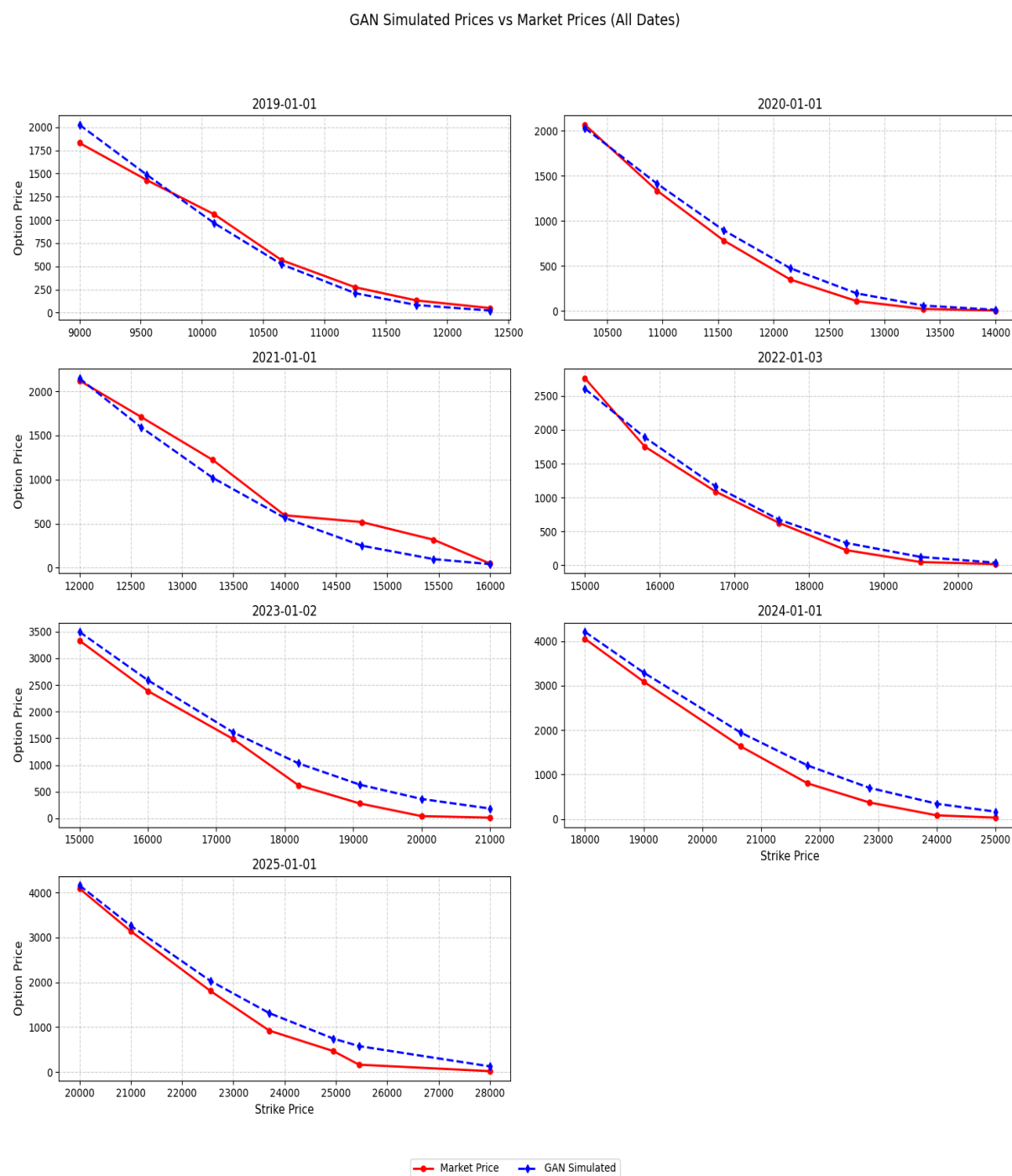


FIGURE 4. GAN Monte Carlo Prices vs Market Prices (Source: Authors' Own).

Calibrated Black–Scholes, using implied volatility, was repeatedly the best-performing model, consistently achieving the lowest MSE across most years, which highlights the powerful predictive quality of implied volatility derived from the market itself. The Heston Model offered competitive results, often placing second; it is plausible that a more finely tuned calibration process, particularly around strikes closer together, could have further improved its accuracy heston. Conversely, the BS Theoretical model, relying solely on historical volatility, performed the worst due to its fundamental inability to capture the volatility smile—the variation of implied volatility across strikes. Notably, the GAN-MC Framework demonstrated strong, comparable performance, achieving the lowest MSE in 2022, with error rates mostly commensurate with the other calibrated models, proving its viability as a data-driven alternative.

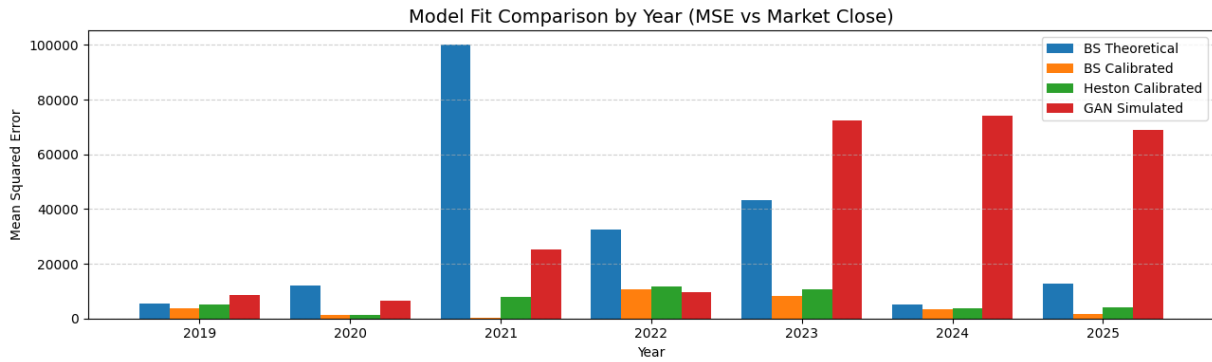


FIGURE 5. Mean Squared Error, All Models (Source: Authors' Own).

Statistical Significance Testing

Given the limited sample size ($n = 7$ option contracts per year), we employ bootstrap resampling and non-parametric hypothesis testing to assess the statistical significance of differences in model performance.

Bootstrap Confidence Intervals for MSE

First, to quantify the uncertainty in our MSE estimates, we apply bootstrap resampling with 10,000 iterations for each model-year combination. For each bootstrap sample, we randomly draw $n = 7$ pricing errors with replacement and compute the MSE. The 2.5th and 97.5th percentiles of the bootstrap distribution provide 95% confidence intervals around the point estimate.

The bootstrap analysis (see Table 1, below) reveals that confidence intervals are generally wide due to the small sample size, indicating substantial uncertainty in MSE point estimates. For instance, in 2019, the GAN model's MSE of 8,547.12 has a 95% confidence interval of [2,225.95, 18,929.29], while BS Calibrated's MSE of 3,888.70 spans [0.00, 11,475.19]. Despite overlapping intervals in some years, certain patterns emerge consistently.

BS Calibrated typically exhibits lower point estimates with confidence bounds that favor superior performance in most years (2019–2021, 2023–2025). Notably, 2022 represents an exception where GAN achieves an MSE of 9,758.94 with a confidence interval of [4,269.10, 16,270.81], competitive with BS Calibrated's 10,670.36 [4.03, 31,231.34] and Heston's 11,689.80 [898.22, 32,018.46]. The relatively tighter confidence bounds for GAN in 2022 suggest more stable performance during that period, consistent with our earlier observation of the GAN's peak pricing accuracy.

Wilcoxon Signed-Rank Tests

Second, to test whether differences in pricing accuracy are statistically significant, we apply the Wilcoxon signed-rank test (see Table 2, below), a non-parametric alternative to the paired t-test that does not assume normality. This test evaluates whether the median absolute pricing errors differ significantly between model pairs. The Wilcoxon tests are conducted both on a year-by-year basis ($n = 7$ per year) and on pooled data across all years ($n = 49$).

When pooling across all years ($n = 49$), the statistical evidence becomes even stronger (see Table 3, below). BS Calibrated significantly outperforms GAN ($W = 47.0, p < 0.001$), as does Heston Calibrated ($W = 129.0, p < 0.001$). The comparison between BS Calibrated and Heston also shows highly significant differences ($W = 0.0, p < 0.001$), with BS Calibrated maintaining a consistent advantage. Finally, the comparison of BS Theoretical versus BS Calibrated yields overwhelming evidence ($W = 4.0, p < 0.001$) that calibration to market-implied volatility is essential for accurate pricing.

TABLE 1. Bootstrap 95% Confidence Intervals for Mean Squared Error (MSE) by Year.

Year	Model	Mean Squared Error	95% CI Lower	95% CI Upper
2019	Black Scholes Theoretical	5,602.24	650.12	13,125.39
2019	Black Scholes Calibrated	3,888.70	0.00	11,475.19
2019	Heston	4,990.19	249.25	12,729.17
2019	GAN	8,547.12	2,225.95	18,929.29
2020	Black Scholes Theoretical	11,948.81	5,096.94	19,372.35
2020	Black Scholes Calibrated	1,167.40	22.64	2,951.00
2020	Heston	1,191.99	42.89	2,959.48
2020	GAN	6,343.17	2,489.72	10,616.51
2021	Black Scholes Theoretical	100,205.35	59,956.74	157,272.49
2021	Black Scholes Calibrated	85.19	0.00	255.57
2021	Heston	8,033.20	4,571.45	12,366.63
2021	GAN	25,174.98	7,103.26	45,566.01
2022	Black Scholes Theoretical	32,580.90	10,236.70	62,844.25
2022	Black Scholes Calibrated	10,670.36	4.03	31,231.34
2022	Heston	11,689.80	898.22	32,018.46
2022	GAN	9,758.94	4,269.10	16,270.81
2023	Black Scholes Theoretical	43,360.64	22,376.57	65,540.27
2023	Black Scholes Calibrated	8,328.07	0.00	18,363.07
2023	Heston	10,556.83	1,876.97	20,582.45
2023	GAN	72,511.70	35,228.48	114,488.78
2024	Black Scholes Theoretical	5,220.91	1,678.67	9,256.60
2024	Black Scholes Calibrated	3,316.11	0.00	7,529.57
2024	Heston	3,583.01	259.08	7,743.98
2024	GAN	74,045.72	40,582.43	110,488.20
2025	Black Scholes Theoretical	12,607.54	4,095.37	22,877.70
2025	Black Scholes Calibrated	1,731.75	0.00	4,066.66
2025	Heston	4,071.45	1,542.73	6,615.71
2025	GAN	69,076.46	25,771.46	118,414.50

Summary Statistics and Interpretation

Aggregate summary statistics across all 49 contracts reveal key performance characteristics (see Table 4, below). BS Calibrated exhibits the lowest mean absolute error (28.17) and remarkably, a median absolute error of 0.00, indicating that for at least half the contracts, the model achieves near-perfect pricing. Heston Calibrated shows a mean absolute error of 56.06 with median of 40.83, while GAN demonstrates a mean absolute error of 156.13 with median of 124.29. BS Theoretical performs worst with mean absolute error of 136.16. The standard deviation of errors for GAN (167.36) is notably higher than BS Calibrated (59.06) and Heston (75.89), suggesting less consistent performance across different market conditions and strike prices.

The statistical tests confirm that while GAN-based pricing is competitive and occasionally achieves comparable MSE (notably in 2022), it does not consistently outperform calibrated parametric models in terms of pricing accuracy. The strength of the GAN approach lies not in point estimate precision but in distributional fidelity—its ability to generate synthetic return paths that capture higher-order moments and tail behavior of the empirical distribution.

TABLE 2. Wilcoxon Signed-Rank Test Results by Year ($n = 7$ contracts per year)

Year	Comparison	Test Statistic		Significance
2019	Black Scholes Calibrated vs GAN	0.0	0.0156	**
2019	Heston vs GAN	0.0	0.0156	**
2019	Black Scholes Calibrated vs Heston	0.0	0.0156	**
2019	Black Scholes Theoretical vs Black Scholes Calibrated	0.0	0.0156	**
2020	Black Scholes Calibrated vs GAN	1.0	0.0312	**
2020	Heston vs GAN	1.0	0.0312	**
2020	Black Scholes Calibrated vs Heston	0.0	0.0625	*
2020	Black Scholes Theoretical vs Black Scholes Calibrated	1.0	0.0312	**
2021	Black Scholes Calibrated vs GAN	0.0	0.0156	**
2021	Heston vs GAN	8.0	0.3750	n.s.
2021	Black Scholes Calibrated vs Heston	0.0	0.0156	**
2021	Black Scholes Theoretical vs Black Scholes Calibrated	0.0	0.0156	**
2022	Black Scholes Calibrated vs GAN	7.0	0.2969	n.s.
2022	Heston vs GAN	9.0	0.4688	n.s.
2022	Black Scholes Calibrated vs Heston	0.0	0.0312	**
2022	Black Scholes Theoretical vs Black Scholes Calibrated	0.0	0.0156	**
2023	Black Scholes Calibrated vs GAN	1.0	0.0312	**
2023	Heston vs GAN	1.0	0.0312	**
2023	Black Scholes Calibrated vs Heston	0.0	0.0156	**
2023	Black Scholes Theoretical vs Black Scholes Calibrated	0.0	0.0156	**
2024	Black Scholes Calibrated vs GAN	0.0	0.0156	**
2024	Heston vs GAN	0.0	0.0156	**
2024	Black Scholes Calibrated vs Heston	0.0	0.0312	**
2024	Black Scholes Theoretical vs Black Scholes Calibrated	0.0	0.0156	**
2025	Black Scholes Calibrated vs GAN	1.0	0.0312	**
2025	Heston vs GAN	1.0	0.0312	**
2025	Black Scholes Calibrated vs Heston	0.0	0.0156	**
2025	Black Scholes Theoretical vs Black Scholes Calibrated	0.0	0.0156	**

Note: Significance levels: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$, n.s. = not significant

TABLE 3. Wilcoxon Signed-Rank Test Results - Pooled Analysis ($n = 49$ contracts)

Comparison	Test Statistic	p -value	Significance
Black Scholes Calibrated vs GAN	47.0	< 0.001	***
Heston vs GAN	129.0	< 0.001	***
Black Scholes Calibrated vs Heston	0.0	< 0.001	***
Black Scholes Theoretical vs Black Scholes Calibrated	4.0	< 0.001	***

TABLE 4. Summary Statistics of Pricing Errors (All 49 Contracts)

Statistic	BS Theoretical	BS Calibrated	Heston	GAN
Mean Error	120.19	27.44	25.71	102.40
Standard Deviation	126.89	59.06	75.89	167.36
Mean Absolute Error	136.16	28.17	56.06	156.13
Median Absolute Error	104.06	0.00	40.83	124.29

Limitations and Interpretation

The limited sample size (7 contracts per year) represents a constraint on statistical power. Year-by-year tests have reduced ability to detect differences, though the consistency of results across multiple years strengthens the conclusions. The statistical significance observed in pooled analysis ($n = 49$) lends credibility to the finding that calibrated models currently maintain a pricing advantage for vanilla options. However, GANs offer complementary value in applications where distributional realism is paramount, such as scenario generation for stress testing, risk management of exotic derivatives, and portfolio simulation under extreme market conditions. The GAN's ability to preserve skewness, kurtosis, and volatility clustering—while simultaneously generating thousands of realistic paths—positions it as a valuable tool for risk assessment rather than as a replacement for established pricing models for standard vanilla options.

Computational Cost and Training:

In terms of computational complexity, the Heston calibration and the GAN training process were similarly time-intensive. Both models averaged around 8–9 minutes per yearly calibration/training window. For the GAN, this calculation excluded the payoff calculation itself. The GAN's performance could likely be significantly enhanced and its computational time reduced with dedicated GPU (Graphics Processing Unit) resources, which were not utilized in this study. The training stability of the Generative Adversarial Network was generally robust. Across most years, the Wasserstein GAN exhibited stable training dynamics, with both the Discriminator (D) and Generator (G) losses converging to near zero.

Distributional Matching:

A central finding is the apparent conflict: the GAN-MC model frequently produces synthetic return distributions that most closely match realized market distributions, yet this superior distributional alignment does not consistently translate into more accurate option prices.

The results of the distributional matching exercise are presented in Table 5, below. Note that we use the terms “BS-Historical” and “BS_Theoretical” as well as “BS-Implied” and “BS_Calibrated” interchangeably. Statistical moments are represented with Greek notation (mean = μ , standard deviation = σ , skewness = γ_1 , kurtosis = γ_2). The actual observed statistical moments for the given time period are denoted with “(A)”.

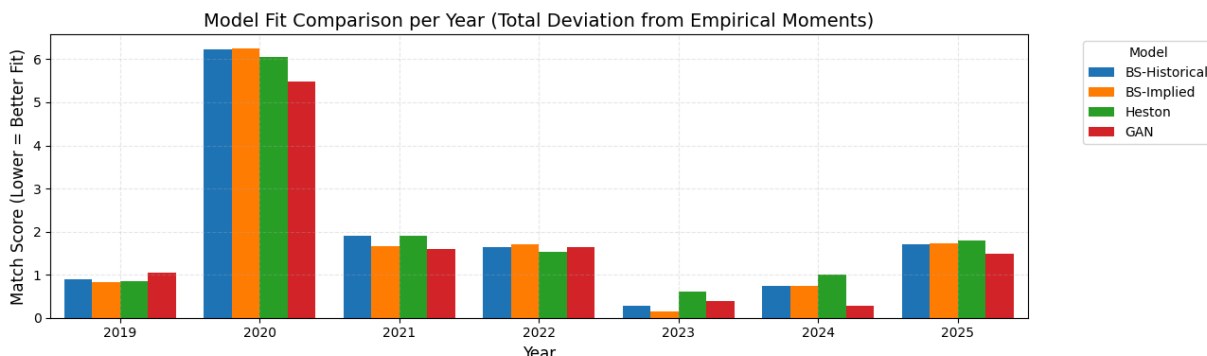


FIGURE 6. Model Fit Comparison per Year (Total Deviation from Empirical Moments) (Source: Authors' Own).

This data is further summarized in Figure 6, above. The GAN achieves best distributional fit in 4 of 7 years, yet does not consistently outperform in pricing. This paradox has three key explanations.

TABLE 5. Distributional Matching Results - Model Moments vs. Realized Market Moments

Year	Model			1						Match
2019	BS_H	-0.00	0.08	0.00	0.03	0.00	0.01	0.18	-0.63	0.90
2019	BS_I	-0.00	0.01	0.00	0.02	0.00	0.01	0.18	-0.63	0.82
2019	Heston	-0.01	0.09	0.03	-0.01	0.00	0.01	0.18	-0.63	0.85
2019	GAN	0.02	0.07	0.27	0.25	0.00	0.01	0.18	-0.63	1.05
2020	BS_H	-0.00	0.08	0.03	0.04	-0.00	0.03	-1.08	5.13	6.24
2020	BS_I	-0.00	0.01	0.01	-0.02	-0.00	0.03	-1.08	5.13	6.26
2020	Heston	-0.00	0.05	0.95	1.11	-0.00	0.03	-1.08	5.13	6.06
2020	GAN	0.01	0.08	-0.39	0.39	-0.00	0.03	-1.08	5.13	5.48
2021	BS_H	-0.01	0.18	0.04	0.04	0.00	0.01	-0.03	1.68	1.90
2021	BS_I	0.00	0.01	-0.02	0.03	0.00	0.01	-0.03	1.68	1.67
2021	Heston	-0.01	0.12	0.19	0.11	0.00	0.01	-0.03	1.68	1.90
2021	GAN	0.01	0.09	-0.23	0.36	0.00	0.01	-0.03	1.68	1.59
2022	BS_H	-0.01	0.09	-0.02	0.05	-0.00	0.01	-0.47	1.17	1.65
2022	BS_I	-0.00	0.01	0.00	-0.06	-0.00	0.01	-0.47	1.17	1.70
2022	Heston	-0.00	0.06	0.86	1.02	-0.00	0.01	-0.47	1.17	1.53
2022	GAN	0.01	0.10	-0.12	-0.03	-0.00	0.01	-0.47	1.17	1.64
2023	BS_H	-0.01	0.10	-0.01	0.02	0.00	0.01	-0.05	-0.11	0.27
2023	BS_I	-0.00	0.01	-0.03	0.02	0.00	0.01	-0.05	-0.11	0.15
2023	Heston	-0.00	0.06	-0.28	0.20	0.00	0.01	-0.05	-0.11	0.61
2023	GAN	0.01	0.12	0.02	0.07	0.00	0.01	-0.05	-0.11	0.38
2024	BS_H	-0.00	0.06	-0.01	-0.01	0.00	0.01	-0.44	0.25	0.74
2024	BS_I	-0.00	0.01	-0.01	-0.06	0.00	0.01	-0.44	0.25	0.74
2024	Heston	-0.00	0.06	0.37	0.11	0.00	0.01	-0.44	0.25	1.01
2024	GAN	0.02	0.11	-0.33	0.21	0.00	0.01	-0.44	0.25	0.27
2025	BS_H	-0.00	0.08	-0.03	0.00	0.00	0.01	-0.33	1.33	1.71
2025	BS_I	-0.00	0.01	0.03	-0.04	0.00	0.01	-0.33	1.33	1.73
2025	Heston	-0.00	0.07	0.15	0.07	0.00	0.01	-0.33	1.33	1.80
2025	GAN	0.02	0.11	-0.14	0.15	0.00	0.01	-0.33	1.33	1.49

Note: Match Score = sum of absolute deviations across mean, std, skewness, and kurtosis. Lower is better.

Why Distributional Matching Does Not Ensure Pricing Accuracy

First, temporal misalignment creates disconnect. GAN learns from historical returns and generates scenarios representing plausible futures, but option prices reflect market expectations before realization. A model can match ex-post distributions while mispricing options if market expectations differ from outcomes. This is evident in 2020, where all models struggled (Match Scores $>$) due to the unanticipated market shock.

Second, vanilla options are insensitive to higher-order moments. European call pricing depends primarily on the first two moments (mean and variance) under the risk-neutral measure. While skewness and kurtosis affect the volatility smile, they have second-order effects. Match Score weights all four moments equally, but pricing does not. BS-Implied often achieves worse Match Scores than GAN but superior pricing (e.g., 2019: Match Score 0.82 vs 1.05, but MSE 3,889 vs 8,547).

Third, specific quantile sensitivity matters more than overall shape. Option payoffs depend on tail probabilities at specific strikes, not full distributional concordance. GAN can match aggregate moments while misestimating tail probabilities relevant for extreme strikes.

Why Implied Volatility Dominates

Market-implied volatility aggregates forward-looking expectations from thousands of participants, incorporating current conditions, news flow, and risk sentiment unavailable in historical data. GAN learns backward-looking patterns; even with drift correction, its volatility structure reflects past regimes. This informational advantage explains why BS-Implied and Heston Calibrated consistently outperform: they use IV as a sufficient statistic capturing all pricing-relevant information.

GAN's Strength: Tail Behavior and Risk Applications

GAN excels at capturing non-Gaussian features. Table 5 shows GAN generates distributions with realistic skewness (-0.39 to 0.27) and kurtosis (0.15 to 0.39), learning volatility clustering and asymmetric shocks. In 2020, GAN achieved a skewness of -0.39 versus BS-Implied's 0.01 , better capturing tail asymmetry (though underestimating the realized -1.08).

This distributional fidelity is valuable for applications beyond vanilla pricing: VaR/CVaR calculations, exotic derivatives (barriers, path-dependent options), regulatory stress testing, and portfolio scenario generation. For vanilla Europeans, however, these higher-order features provide limited value over correctly specified implied volatility.

Statistical Analysis: Visualizations and Robustness Check

The four panels in Figure 7 below provide complementary perspectives on model performance. The top-left panel displays the bootstrap distribution of MSE for 2022, the year where GAN achieved its most competitive pricing performance. The overlapping distributions illustrate why statistical tests failed to detect significant differences between models that year—the bootstrap samples show substantial overlap, particularly between GAN (red) and the calibrated models (blue and green).

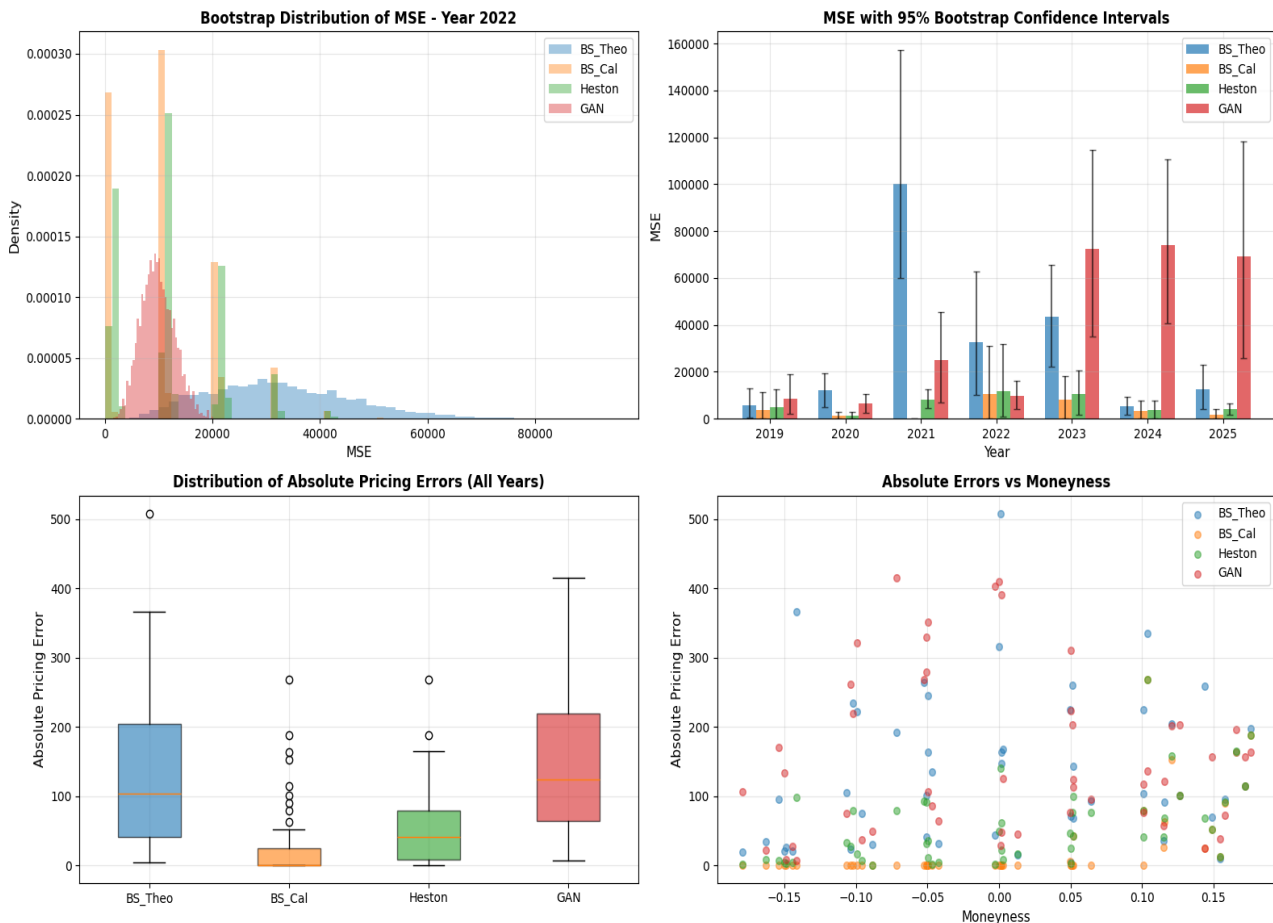


FIGURE 7. Bootstrap Distribution of MSE, MSE Plot with Confidence Intervals, Box and Whisker Plot of Errors Across Years, Robustness of Model Performance Across Moneyness
Source: Authors' Own

The top-right panel presents MSE point estimates with 95% bootstrap confidence intervals across all years, visually confirming the wide uncertainty bands discussed earlier and highlighting 2022 as an outlier where GAN's confidence interval is notably tighter and lower than in other years. The bottom-left panel shows the distribution of absolute pricing errors pooled across all 49 contracts, with GAN exhibiting a substantially wider spread and higher median than BS Calibrated or Heston, consistent with the summary statistics in Table 4.

The bottom-right panel reveals that GAN's pricing errors are relatively uniform across moneyness levels, whereas BS Calibrated achieves near-zero errors for at-the-money options (moneyness ~ 0) but larger errors for deep in-the-money and out-of-the-money contracts. This suggests that while calibrated models excel at pricing near-ATM options where liquidity is highest, GAN's errors are more evenly distributed across the volatility smile, though consistently larger in magnitude.

Interpretation

The GAN successfully learns complex return characteristics but cannot replicate forward-looking market information embedded in implied volatility. Future work could explore conditional GANs incorporating current market state (VIX, IV skew) to bridge this gap, or position GANs as complementary tools: calibrated models for vanilla pricing, GAN scenarios for risk management and exotic derivatives where distributional realism is paramount.

5|Conclusion

The hybrid GAN–Monte Carlo framework offers a flexible and adaptive approach to vanilla option pricing. It overcomes the rigidity of parametric models and enhances scenario diversity. However, GANs require careful tuning to avoid overfitting and mode collapse. Future work could explore conditional GANs for implied volatility surfaces or integrate reinforcement learning for hedging strategies.

All experiments were conducted on a CPU (Central Processing Unit) only environment, which likely underestimates the performance of the GAN. With GPU (Graphics Processing Unit) acceleration, the training time would be expected to decrease substantially, especially during backpropagation through large batch iterations [32, 33].

This paper demonstrates the viability of GANs for scenario generation in vanilla option pricing. By combining deep generative modeling with classical simulation, the proposed framework occasionally improves pricing accuracy and robustness in emerging markets. In particular, given the distributional fidelity of the GAN Monte Carlo simulations, the approach is extensible to other asset classes and may support broader applications in financial risk modeling as well as AI-driven trading systems.

Data Sources

NIFTY 50 Historical Data: Yahoo Finance (ticker: $\hat{N}SEI$), daily closing prices from January 1, 2015 to August 31, 2025 [28].

Option Market Data: National Stock Exchange of India (NSE), European-style call options with approximately 3-month maturities [29].

Risk-Free Rate: Indian 3-Month Treasury Bill Yield [30].

Code Availability

The complete implementation code, including GAN training, risk-neutral drift correction, and Monte Carlo pricing modules, is available upon request from the corresponding author.

Acknowledgments

The authors express their sincere gratitude to the editors and anonymous reviewers for their invaluable comments and constructive feedback, which significantly contributed to the enhancement of this paper. The authors also acknowledge the use of Large Language Models (LLMs) for assistance with code generation and writing review; however, the intellectual endeavour, analysis, and conclusions are solely those of the authors.

Author Contribution

Anand Sharma: conceptualization, methodology, software, data curation, formal analysis, writing, supervision. Danny Seow Wei Jie: resources, writing, project administration. All authors have read and agreed to the published version of the manuscript.

Funding

The authors declare that no external funding or support was received for the research presented in this paper, including administrative, technical, or in-kind contributions.

Conflicts of Interest

The authors declare that there is no conflict of interest concerning the reported research findings.

References

- [1] Choe, G. H. (2016). Pricing of vanilla options. In *stochastic analysis for finance with simulations*. Springer. https://doi.org/10.1007/978-3-319-25589-7_17
- [2] Wang, L., Zhang, M., & Liu, Z. (2023). The progress of Black-Scholes model and Black-Scholes-Merton model. *BCP business & management*, 38, 3405-3410. <https://doi.org/10.54691/bcpbm.v38i.4314>
- [3] Rockinger, M., & Jondeau, É. (2003). Conditional volatility, skewness, and kurtosis. *Journal of economic dynamics and control*, 27(10), 1699-1737. [https://doi.org/10.1016/S0165-1889\(02\)00079-9](https://doi.org/10.1016/S0165-1889(02)00079-9)
- [4] Matsakos, T., & Nield, S. (2024). Quantum Monte Carlo simulations for financial risk analytics. *Quantum*, 8, 1306. <https://doi.org/10.22331/q-2024-04-04-1306>
- [5] Sharma, P., Kumar, M., Sharma, H. K. (2024). Generative adversarial networks (GANs). *Multimedia tools and applications*, 83, 88811-88858. <https://doi.org/10.1007/s11042-024-18767-y>
- [6] Heston, S. L. (1993). A closed-form solution for options with stochastic volatility. *The review of financial studies*, 6(2), 327-343. <https://doi.org/10.1093/rfs/6.2.327>
- [7] 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>
- [8] Merton, R. C. (1973). The theory of rational option pricing. *The bell journal of economics*, 4(1), 141-183. https://doi.org/10.1142/9789812701022_0008
- [9] Chang, H. (2024). Stochastic volatility models and their applications to financial markets. *Advances in economics, management and political sciences*, 135, 103-109. <https://doi.org/10.54254/2754-1169/2024.18738>
- [10] Gatheral, J. (2006). *The volatility surface: A practitioner's guide*. John Wiley & Sons. <https://doi.org/10.1002/9781119202073>
- [11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), 139-144. <https://doi.org/10.1145/3422622>
- [12] Eckerli, F., & Osterrieder, J. (2021). Generative adversarial networks in finance: An overview. <https://doi.org/10.2139/ssrn.3864965>
- [13] Eckerli, F., & Osterrieder, J. (2021). Generative adversarial networks in finance: An overview. <https://doi.org/10.48550/arXiv.2106.06364>
- [14] Athey, S., Imbens, G. W., Metzger, J., & Munro, E. (2024). Using Wasserstein generative adversarial networks for the design of Monte Carlo simulations. *Journal of econometrics*, 240(2), 105076. <https://doi.org/10.1016/j.jeconom.2020.09.013>
- [15] De Prado, M. L. (2018). *Advances in financial machine learning*. John Wiley & Sons. <https://www.wiley.com/en-ie/shop/general-finance-investments/advances-in-financial-machine-learning-p-9781119482086>
- [16] Jabbar, A., Li, X., & Omar, B. (2021). A survey on generative adversarial networks. *ACM computing surveys*, 54, 1-49. <https://doi.org/10.1145/3463475>

- [17] Yoon, J., Jarrett, D., & Van Der Schaar, M. (2019). Time-series generative adversarial networks. *advances in neural information processing systems* (pp. 1-11). Neural Information Processing Systems Foundation. <https://papers.nips.cc/paper/2019/file/c9efe5f26cd17-ba6216bbe2a7d26d490-Paper.pdf>
- [18] Chen, S., Ren, S., & Zhang, Q. (2025). Hybrid architectures that combine LLMs and predictive analytics for next-generation financial modeling. *Mathematical modeling and algorithm application*, 6, 31-43. <https://doi.org/10.54097/j1cjb453>.
- [19] Lamina, O. (2023). A hybrid predictive modeling framework for financial risk assessment. *International journal of science and research archive*, 10, 1236-1245. <https://doi.org/10.30574/ijrsra.2023.10.1.0748>
- [20] Wiese, M., Knobloch, R., Korn, R., & Kretschmer, P. (2020). Quant GANs: Deep generation of financial time series. *Quantitative finance*, 20(9), 1419-1440. <https://doi.org/10.1080/14697688.2020.1730426>
- [21] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3), 307-327. [https://doi.org/10.1016/0304-4076\(86\)90063-1](https://doi.org/10.1016/0304-4076(86)90063-1)
- [22] Aggarwal, A., Mittal, M., & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. *International journal of information management data insights*, 1(1), 100004. <https://doi.org/10.1016/j.jjime.2020.100004>
- [23] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein GANs. *Proceedings of the 31st international conference on neural information processing systems* (pp. 5769-5779). Curran Associates Inc. <https://doi.org/10.5555/3295222.3295327>
- [24] Van Dijk, M. T. P., De Graaf, C. S. L., & Oosterlee, C. W. (2018). Between P and Q. *Journal of risk and financial management*, 11(4), 67. <https://doi.org/10.3390/jrfm11040067>
- [25] Hossain, A. S., Noushin, M. T., & Hasan, K. (2019). A comparison of the Black-Scholes option pricing model and its alternatives. *Dhaka university journal of science*, 67(2), 105-110. <https://www.banglajol.info/index.php/DUJS/article/view/54581/38354>
- [26] Storn, R., & Price, K. (1997). Differential evolution. *Journal of global optimization*, 11(4), 341-359. <https://doi.org/10.1023/A:1008202821328>
- [27] Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM journal on scientific computing*, 16(5), 1190-1208. <https://doi.org/10.1137/0916069>
- [28] Yahoo Finance. (2025). Historical data for Nifty 50 [data set]. <https://finance.yahoo.com/quote/%5ENSEI/history/?guccounter>
- [29] National Stock Exchange of India. (2026). NIFTY 50 option market data [data set]. <https://www.nseindia.com/>
- [31] Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? *IEEE signal processing magazine*, 26(1), 98-117. <https://doi.org/10.1109/MSP.2008.930649>.
- [32] Hasnaine, Q. R., Fouda, M. M., & Chiu, S. C. (2025, April). GPU vs. CPU: A comparative study of performance, architecture, and NVIDIA's innovations in modern computing. In *2025 IEEE 4th international conference on computing and machine intelligence (ICMI)* (pp. 1-6). IEEE. <https://doi.org/10.1109/ICMI65310.2025.11141315>
- [33] Raj, P., & Sekhar, C. (2020). Comparative study on CPU, GPU and TPU. *international journal of computer science and information technology for education*, 5, 31-38. <https://doi.org/10.21742/IJCSITE.2020.5.1.04>.

6|Appendix

This appendix provides comprehensive implementation details to facilitate replication of the GAN-based option pricing framework. All experiments were conducted in a CPU-only environment without GPU acceleration.

Computational Environment

The experiments were performed on the following hardware and software configuration:

Operating System: Windows 10 (64-bit)
 Processor: Intel Core i5-10350U CPU @ 1.20 GHz
 RAM: 8 GB
 Python Version: 3.12.12
 TensorFlow Version: 2.19.0
 NumPy Version: 2.0.2
 Computing Environment: Google Colab (CPU runtime)

Model Hyperparameters

The Wasserstein GAN architecture and training procedure employed the following hyperparameters:

Latent Dimension: 32
 Training Epochs: 400 per year
 Batch Size: 32
 Learning Rate: 0.0001 (both generator and discriminator)
 Optimizer: Adam (default parameters: $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \cdot 10^{-8}$)
 Discriminator Updates per Generator Update: 2 ($n_{\text{critic}} = 2$)
 Weight Clipping Range: $[-0.01, 0.01]$
 Number of Synthetic Paths Generated: 10,000 per year

Generator Architecture

Input Layer: Latent vector of dimension 32
 Hidden Layer: Fully connected, 64 neurons, ReLU activation
 Output Layer: Fully connected, T neurons (no activation), where $T \in \{83, 85, 86, 87\}$
 Reshape: Output reshaped to $(T, 1)$

Discriminator Architecture

Input Layer: Sequence of shape $(T, 1)$, Output Layer: Single neuron (no activation)
 Convolutional Layer: 32 filters, kernel size 3, same padding, ReLU activation
 Flatten Layer: Convert 2D feature maps to 1D
 Hidden Layer: Fully connected, 32 neurons, ReLU activation

Data Preprocessing

Historical NIFTY 50 log-returns were normalized using z-score standardization before training:

$$r_{\text{norm}} = \frac{r - \mu_{\text{train}}}{\sigma_{\text{train}}} \quad (\text{A1})$$

where μ_{train} and σ_{train} are computed from all available training data up to (but excluding) each target year. After generation, synthetic returns were denormalized and scaled to match the empirical volatility of the training data, followed by risk-neutral drift correction as described in Section 3.4.

Training Dataset Sizes

The expanding window training approach resulted in the following dataset sizes per year.

TABLE A1. Training Dataset Sizes per Yearly Window

Year	Sequence Length (T)	Training Observations	Training Sequences
2019	86 days	1,000	914
2020	85 days	1,250	1,165
2021	83 days	1,500	1,417
2022	87 days	1,750	1,663
2023	86 days	2,000	1,914
2024	87 days	2,250	2,163
2025	85 days	2,500	2,415

Training Stability and Convergence

WGAN training exhibited stable convergence across all years. Both discriminator and generator losses approached zero as training progressed, indicating successful adversarial equilibrium. Representative training dynamics are shown below for selected epochs:

TABLE A2. WGAN Loss Convergence by Epoch

Year	Epoch 0	Epoch 100	Epoch 200	Epoch 300
2019		= 005 = 003	= 007 = 007	= 010 = 012
2020		= 004 = 003	= 006 = 008	= 008 = 009
2021		= 002 = 002	= 005 = 006	= 006 = 004
2022		= 002 = 002	= 002 = 004	= 004 = 004
2023		= 003 = 003	= 004 = 004	= 007 = 007
2024		= 004 = 004	= 006 = 008	= 016 = 013
2025		= 005 = 004	= 008 = 007	= 012 = 012

No mode collapse or training instability was observed. Losses converged smoothly without divergence or oscillation, confirming the robustness of the WGAN framework with weight clipping.

Computational Efficiency

Total training and path generation time across all seven years was approximately 8.37 minutes (CPU-only). Individual year training times ranged from approximately 1.0 to 1.3 minutes per model. This excludes subsequent option pricing calculations via Monte Carlo simulation.

Random Seed and Reproducibility

No explicit random seed was set in the original experiments reported in this paper. Consequently, exact numerical replication will produce different (but statistically similar) results due to stochastic initialization of network weights and random sampling during training.

For deterministic reproducibility in future work, researchers should add the following code at the beginning of their implementation:

```
import numpy as np
import tensorflow as tf
RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
```