big data and cognitive computing

MDPI

Article

Bio-Inspired Spiking Recurrent Networks with Evolutionary Optimization for Non-Stationary Cryptocurrency Forecasting

Francis Noah Walugembe [1,*,†], Maciej Wielgosz [2,†], Matej Mertik [1,†] and Matjaž Gams [3,†]

[1] Department of Applied Artificial Intelligence, Alma Mater Europaea University, Slovenska Street 17, 2000 Maribor, Slovenia; matej.mertik@almamater.si
[2] Faculty of Computer Science, Electrical and Telecommunications, AGH University of Krakow, 30-059 Krakow, Poland; wielgosz@agh.edu.pl
[3] Department of Intelligent Systems, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia; matjaz.gams@ijs.si
* Correspondence: francis.walugembe@almamater.si; Tel.: +386-256758950945
† These authors contributed equally to this work.

Abstract

Forecasting cryptocurrency prices remains difficult because market dynamics are highly volatile, non-stationary, and regime-dependent. This study investigates whether combining a spiking-inspired recurrent architecture with the Grey Wolf Optimizer (GWO) can improve one-step-ahead Bitcoin forecasting within a controlled model family. We compare four configurations, LSTM, SLSTM, GWO-LSTM, and GWO-SLSTM, on 4039 daily BTC–USD closing prices from 17 September 2014 to 9 October 2025 using Min–Max normalization, strict chronological splitting, windowed regime-based robustness analysis across three distinct market regimes, and repeated-run testing. The proposed SLSTM replaces the conventional hidden-state recurrence with leaky integrate-and-fire-inspired synaptic, membrane, and adaptive-threshold dynamics, functioning as a spiking-inspired recurrent model with thresholded event gating (reset = 'none', learnable threshold). On the primary hold-out split, GWO-SLSTM achieved a test RMSE of 1840.97 and a test MAPE of 1.76%, compared with 2217.24 and 2.46% for GWO-LSTM, 3501.48 and 3.86% for SLSTM, and 4030.10 and 4.40% for LSTM. Both GWO-optimized models exhibited substantial improvements over their non-optimized counterparts, while the SLSTM baseline also outperformed the plain LSTM, indicating gains from both spiking-inspired recurrence and evolutionary hyperparameter optimization. Both optimized models exhibited near-zero bias (PBIAS 0.11% for GWO-LSTM and 0.36% for GWO-SLSTM). Within the present implementation, GWO-SLSTM also trained faster than GWO-LSTM (39.71 s vs. 137.28 s), although this runtime difference should be interpreted as setup-specific because the model families were implemented in different frameworks and stopped after different numbers of epochs. Overall, within the expanded univariate BTC–USD setting, the results support GWO-SLSTM as a strong within-family candidate for one-step-ahead forecasting under non-stationary conditions.

Keywords: bitcoin; time-series forecasting; spiking neural networks; LSTM; grey wolf optimizer; surrogate gradient; evolutionary algorithms

Academic Editors: Cuong Nguyen and Bo Qin

Received: 18 March 2026
Revised: 28 May 2026
Accepted: 3 June 2026
Published: 23 June 2026

1. Introduction

Forecasting cryptocurrency prices remains a challenging task due to extreme volatility, non-stationarity, and abrupt regime shifts that limit the robustness and generalization

ability of conventional forecasting models. Traditional time series approaches, including ARIMA, GARCH, and support vector regression, rely on assumptions of linearity and stationarity that are frequently violated in cryptocurrency markets, resulting in limited predictive performance under rapidly changing conditions [1,2].

Recurrent neural networks, and, in particular, Long Short-Term Memory (LSTM) architectures, have become a standard choice for financial time-series forecasting due to their ability to model nonlinear temporal dependencies and long-range dynamics. LSTMs have demonstrated strong performance across a range of financial prediction tasks, including stock indices and cryptocurrency prices [3,4]. However, their effectiveness is highly sensitive to hyperparameter selection, and they are prone to overfitting and unstable convergence when exposed to noisy or non-stationary data [5,6]. To mitigate these issues, metaheuristic optimization techniques such as the Grey Wolf Optimizer (GWO) have been introduced to automate hyperparameter tuning and improve convergence reliability [3].

Spiking-inspired recurrent architectures, based on leaky integrate-and-fire (LIF) mechanisms, represent a biologically motivated computational paradigm that processes information through discrete spike events governed by membrane dynamics. By incorporating leaky integrate-and-fire mechanisms and event-driven computation, SNNs offer high temporal resolution, sparse activation, and potential energy efficiency, making them well suited for non-stationary and resource-constrained settings [7,8]. Despite these advantages, training SNNs remains challenging due to the non-differentiability of spike functions. Surrogate gradient methods have been proposed to enable gradient-based learning, but stability and scalability issues persist, particularly for long temporal sequences [9,10]. Moreover, the integration of spiking dynamics into recurrent architectures suitable for financial forecasting remains largely unexplored.

Motivated by these observations, this paper investigates whether spiking, leaky integrate-and-fire-inspired state dynamics can improve one-step-ahead Bitcoin forecasting under non-stationary conditions when compared with a matched LSTM baseline under identical optimization budgets. The contribution is methodological rather than scale-driven: we evaluate a hybrid recurrent design in which the conventional hidden-state update is replaced by spiking-inspired membrane dynamics, while the learning rate and weight decay are selected by the Grey Wolf Optimizer (GWO) for both models under the same search budget. This framing helps separate architectural and optimization effects and supports a reproducible comparison under strict chronological validation.

The main contributions of this work are summarized as follows:

- A spiking-inspired recurrent model (SLSTM) that embeds LIF-inspired synaptic, membrane, and adaptive-threshold dynamics with thresholded event gating (`reset = 'none'`, learnable threshold) into an LSTM-like update for one-step-ahead univariate BTC–USD forecasting, together with Grey Wolf Optimizer (GWO)-based tuning of learning rate and weight decay. The overall contribution is framed as a controlled within-family comparison.
- A four-configuration comparative design comprising LSTM, SLSTM, GWO-LSTM, and GWO-SLSTM. This design enables us to separate the architectural effect of spiking recurrence, the optimization effect of GWO within each architecture, and the combined effect under matched data splits, search budgets, and evaluation metrics.
- A reproducible evaluation protocol for regime-shifting financial data based on chronological splitting, windowed regime-based robustness analysis, and repeated-run statistical testing.

The empirical scope is therefore a controlled within-family comparison rather than an exhaustive benchmark against all contemporary forecasting models.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature and positions the proposed approach within prior work, while Section 3 describes the data, preprocessing steps, model formulations, optimization procedure, and evaluation protocol. Section 4 presents the experimental setup; Section 5 reports the empirical results, including hold-out, windowed regime-based robustness, statistical, and computational analyses; and Section 6 discusses the findings, their practical implications, and the main limitations of the study. Finally, Section 7 concludes the paper. Additional descriptive diagnostics are provided in Appendix A, while the nomenclature and core pseudocode are provided in Appendix B

2. Literature Review

This section reviews the three research streams most directly relevant to the present study: deep recurrent forecasting models, evolutionary hyperparameter optimization, and Spiking Neural Networks. It then identifies the specific methodological gap addressed by the proposed approach and summarizes representative prior studies in a structured comparative table.

2.1. Deep Learning Architectures in Financial Forecasting

Neural approaches to financial forecasting evolved from shallow models to recurrent architectures that better capture nonlinear temporal dependencies. Vanilla RNNs are limited by vanishing/exploding gradients on long sequences, and LSTM architectures became a standard solution for modeling longer temporal dependencies in financial series [11]. LSTMs have since been widely adopted for market prediction, with evidence of improved performance across equities and commodity-related forecasting settings [12,13], as well as in cryptocurrency prediction tasks, where LSTM-based methods can outperform classical baselines such as ARIMA [14]. However, practical performance remains sensitive to hyperparameter choices and may degrade under noisy, non-stationary market dynamics, where overfitting and unstable convergence are common [5,6].

Recent work further supports deep sequential models in complex regimes, including multivariate financial forecasting and high-frequency, non-stationary trading scenarios [15,16]. In parallel, automated tuning has become increasingly common; for example, optimization-driven designs such as PSO-tuned deep ConvLSTM variants have been reported to improve convergence reliability and predictive accuracy relative to manual selection [17]. Collectively, these studies motivate robust temporal architectures paired with adaptive hyperparameter selection.

Transformer-based forecasting models have also become an important reference point in recent time-series prediction research. In the general forecasting literature, models such as Informer were developed to improve long-range dependency modeling and computational efficiency, especially in long-context settings [18]. In financial forecasting, transformer-style models have likewise been applied to stock movement prediction, where they provide an alternative inductive bias to conventional recurrent architectures and can capture broader temporal interactions [19]. At the same time, these models often involve higher architectural complexity and are commonly evaluated in richer multivariate settings than the controlled univariate design used here. Expanding on these studies, the present study positions GWO-SLSTM as a more compact recurrent alternative with stronger temporal locality, intended for a controlled univariate and regime-shifting forecasting setup rather than as a direct empirical replacement for transformer-based forecasters.

2.2. Evolutionary and Swarm-Based Optimization in Time Series

Hyperparameter selection is a key bottleneck in deep time-series forecasting because search spaces are non-convex and parameter interactions can be strong. As alternatives to grid and random search, evolutionary and swarm-based methods provide population-level exploration and are frequently used to tune neural forecasting pipelines. Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs), including hybrid formulations for multi-asset settings [20], PSO-based optimization of deep models [21], and modified PSO variants for LSTM hyperparameter tuning in financial series [22], have been applied to financial modeling and deep-learning parameter tuning.

The Grey Wolf Optimizer (GWO) [23] is appealing due to its simple update rules and empirically stable convergence behavior; it has been used in domains such as energy demand forecasting [24] and stock prediction [25], and is often competitive with other metaheuristics. Broader metaheuristic comparisons for LSTM tuning further confirm the value of such approaches [26]. Recent BDCC-oriented studies likewise emphasize swarm intelligence and hybrid evolutionary strategies for improving generalization and robustness in volatile environments [27,28]. These results support using low-budget metaheuristic search to reduce manual tuning and improve training reliability under distribution shifts.

2.3. Spiking Neural Networks and Surrogate Gradients

Spiking Neural Networks (SNNs) model computation via discrete spike events governed by membrane dynamics, offering temporally precise, and potentially sparse, event-driven processing. Classical perspectives characterize SNNs as a biologically grounded neural computation paradigm [29]. However, training is challenging because spike generation is non-differentiable. Surrogate gradient learning addresses this by using differentiable approximations for the spike derivative during backpropagation [30]. Prior work shows that recurrent SNN formulations with gating mechanisms can perform effectively when tackling temporal learning problems [31], and broader surveys highlight event-driven efficiency and the practical role of surrogate gradients in deep SNN training [32]. Nevertheless, most empirical validation is targeted at neuromorphic benchmarks (e.g., vision or speech), and financial time-series forecasting remains comparatively underexplored.

Emerging work has begun connecting event-driven neural computation to data-intensive decision systems [33], while recent studies and reviews highlight the relevance of bio-inspired and cognitive computing approaches for scalable analytics [34,35]. Complementary evidence from deployed AI systems further underlines the importance of rigorous evaluation and robustness under distribution shifts [36]. In this broader context, integrating big-data analytics with cognitive computing paradigms in cloud/IoT environments has been highlighted as a pathway to scalable real-world adoption [37]. These directions motivate investigations of spiking recurrent dynamics for non-stationary financial forecasting, especially when paired with controlled optimization and validation protocols.

2.4. Research Gaps and Motivation

Despite progress in deep forecasting, swarm-based optimization, and spiking computation, their combined use for volatile financial time series remains limited. Three gaps are particularly relevant:

1. Spiking neuron dynamics have been only minimally investigated within LSTM-style architectures for financial forecasting.
2. Joint treatment of conventional training hyperparameters and spiking-specific dynamics within a unified optimization-and-learning pipeline remains largely unexplored.
3. Controlled, chronologically rigorous evaluations of spiking-inspired LSTM-style recurrent models under non-stationary cryptocurrency market regimes remain limited.

Existing studies typically emphasize either hyperparameter tuning of conventional recurrent or transformer-based forecasters, or spiking/event-driven dynamics outside controlled financial forecasting settings; evidence on their combination under a matched, chronologically evaluated financial design remains limited.

2.5. Comparative Summary of Prior Models

Table 1 provides a structured overview of representative studies cited in Sections 2.1–2.3, highlighting their architectures, optimization methods, data regimes, forecasting settings, strengths, and limitations. The table enables a direct visual comparison and suggests that, within the representative literature reviewed here, directly comparable evaluations combining spiking-inspired LSTM-style recurrence, evolutionary hyperparameter optimization, and rigorous chronological validation for volatile financial time series remain limited. This is the specific controlled setting evaluated in the present study.

Table 1. Comparison of prior models relevant to financial time-series forecasting, organized by architecture and optimization strategy. The proposed GWO-SLSTM is included in the last row to indicate the specific controlled comparison evaluated in this study.

Study/Model	Architecture	Optimization Method	Data Regime	Forecasting Setting	Strengths and Shortcomings
Hochreiter and Schmidhuber (1997) [11]	LSTM	Gradient descent (BPTT)	Various sequential benchmarks	General time series	**Strengths:** Solves vanishing gradient. **Shortcomings:** Manual tuning; no spiking.
Cryptocurrency LSTM studies [14]	LSTM	Adam/SGD	Bitcoin, altcoin price data	Price prediction	**Strengths:** Outperforms ARIMA. **Shortcomings:** Sensitive to non-stationarity; no auto-tuning.
PSO-tuned ConvLSTM [17]	ConvLSTM	Particle Swarm Optimization (PSO)	Multivariate financial series	Volatility/price forecasting	**Strengths:** Improves convergence. **Shortcomings:** No spiking; PSO may be costly.
PSO/GA for financial deep learning [20,21]	Generic deep nets (MLP/RNN)	PSO, Genetic Algorithms	Stock indices, equities	Return prediction	**Strengths:** Population exploration. **Shortcomings:** No LSTM gating; small data.
Modified PSO-LSTM [22]	LSTM	Modified PSO	Energy/financial time series	Load/price forecasting	**Strengths:** Tailored PSO. No event-driven computation.
GWO for energy/stock [24,25]	LSTM/ELM	Grey Wolf Optimizer	Electricity demand, stock indices	Demand/price prediction	**Strengths:** Simple, stable convergence. **Shortcomings:** Conventional RNN; no spiking.
Metaheuristic tuning survey [26]	LSTM	Various (GA, PSO, GWO, etc.)	Multiple benchmarks	General time series	**Strengths:** Comprehensive comparison. **Shortcomings:** No new model; no spiking.
SNN surrogate gradient [30–32]	Spiking Neural Networks (feedforward or recurrent)	Surrogate gradient + BPTT	Neuromorphic (vision, speech)	event recognition	**Strengths:** Event-driven efficiency. **Shortcomings:** Rarely financial; no evolutionary tuning.
Transformer-based forecasting studies [18,19]	attention-based models	Gradient-based training	Typically multivariate, longer-context financial series	Price/return forecasting	**Strengths:** Long-range dependency modeling; multivariate interaction learning. **Shortcomings:** Often data- and compute-intensive.
GWO-SLSTM (proposed)	Spiking-inspired LSTM-style recurrent model with learnable threshold and `reset = 'none'`	Grey Wolf Optimizer + early stopping	Bitcoin USD (2014–2025), chronological 80/20 split plus windowed regime-based robustness analysis	Price (regression)	**Strengths:** Combines spiking-inspired recurrence with GWO-based hyperparameter tuning in a controlled forecasting setup. Evaluated only on a univariate BTC–USD series; no stronger external forecasting baseline is included in the present study.

As summarized in Table 1, existing works either apply swarm/evolutionary optimization to conventional recurrent models, develop transformer-based forecasters with different inductive biases and computational trade-offs, or explore spiking neural networks mainly on non-financial benchmarks without evolutionary hyperparameter tuning. Within the representative literature reviewed here, we did not identify a directly comparable study that evaluates a spiking-inspired LSTM-style recurrent model with learnable thresholds and a deliberate `reset = 'none'` configuration together with matched GWO-based tuning for volatile cryptocurrency forecasting. Accordingly, the contribution of the present work is best understood as a controlled within-family comparison under a fixed univariate BTC–USD setup, matched GWO budgets, chronological hold-out testing, and windowed regime-based robustness analysis, rather than as a general claim of superiority over all contemporary forecasting models.

3. Materials and Methods

This section describes the empirical design of the study. It first presents the data collection and preprocessing procedure, then explains the reproducibility controls, model architectures, optimization method, and evaluation protocol used to ensure a fair comparison across the four forecasting configurations.

3.1. Data Collection and Preprocessing

Daily Bitcoin closing prices (BTC–USD) were obtained from Yahoo Finance (https://finance.yahoo.com/quote/BTC-USD/history/?p=BTC-USD) (accessed on 18 January 2026) for the period 17 September 2014 to 9 October 2025 and downloaded via the `yfinance` API (`ticker = 'BTC-USD'`).The access date format is ISO 8601 (also known as "YYYY-MM-DD" or "dash-separated year-month-day"). The resulting dataset contains 4039 consecutive daily observations and spans materially different market phases, including the early market-formation period, the 2017 bull run, the 2018–2019 bear market, the 2020–2021 expansion phase, and the more recent high-volatility correction periods. This broader horizon was selected to move beyond the narrower 2014–2017 proof-of-concept segment used only during preliminary development and not in the final reported experiments, providing a more credible basis for evaluating forecasting robustness under regime shifts and non-stationarity.

Because Bitcoin trades continuously, the series contained no structural weekend or holiday gaps. Prices were scaled to $[0, 1]$ using Min–Max normalization, with scaling parameters fitted on the training portion only and then applied, unchanged, to validation and test segments. A strict chronological split was applied to avoid look-ahead bias: the first 3231 observations (80%) were used for training/validation, and the last 808 observations (20%) were held out for testing. The small fraction of missing values (<0.001%) was handled using linear interpolation.

To assess robustness beyond a single hold-out split, we additionally defined a windowed regime-based robustness analysis in which pre-specified forward-test periods were compared against the available prediction horizon while preserving strict chronology. Three rolling-origin windows were selected to represent materially different market conditions, namely a high-volatility correction/bear-transition segment, a recovery and early expansion segment, and a sustained expansion segment. This design allows the relative model ranking to be examined across temporally distinct regimes rather than relying only on the primary hold-out split. The exact window boundaries and their dominant market characteristics are reported in Table 2 in the main text.

Table 2. Rolling-origin windows and dominant market characteristics used in the robustness analysis. Each window contains a sequential training period followed by a forward-test period to evaluate model generalization under changing Bitcoin market conditions.

Window	Training Period	Forward-Test Period	Dominant Market Character
R1	17 September 2014–6 May 2021	7 May 2021–23 July 2023	High-volatility correction/bear transition
R2	26 October 2015–14 June 2022	15 June 2022–30 August 2024	Recovery and early expansion phase
R3	15 May 2016–2 January 2023	3 January 2023–20 March 2025	Sustained expansion (bull market)

Daily returns R_t were computed to characterize market regimes as bull ($R_t > 1\%$), bear ($R_t < -1\%$), or neutral. Descriptive statistics and stationarity/dependence diagnostics (ADF and Ljung–Box) for the train and test splits are reported in Appendix A (Table A1), together with the regime distribution (Figure A1).

3.2. Rolling-Origin Evaluation

The rolling-origin robustness analysis was designed to assess whether the model ranking observed in the primary chronological hold-out split remains stable across temporally distinct Bitcoin market regimes. Three forward-test windows were pre-specified, as reported in Table 2: R1 represents a high-volatility correction/bear-transition phase, R2 represents a recovery and early expansion phase, and R3 represents a sustained expansion phase. Because the stored prediction series from the final experiment covers the primary 20% hold-out period, quantitative scores are available only for the portions of R2 and R3 that overlap with this prediction horizon. R1 is retained in the window definition table for transparency, but it is not scored in the current quantitative comparison. The reported analysis should be interpreted as a windowed hold-out robustness check over the available prediction horizon, rather than as a complete rolling-origin retraining benchmark across all three windows.

3.3. Rolling-Origin Timeline Visualization

Figure 1 presents a Gantt-style timeline of the training and forward-test periods for the three representative rolling-origin windows (R1, R2, R3). The blue bars indicate the training period (approximately 60% of historical data), while the salmon bars show the forward-test period (the subsequent 20%). The dominant market character for each test window is annotated alongside the bar. This visualization complements Table 2 by illustrating the temporal placement and relative lengths of the windows used in the robustness analysis.

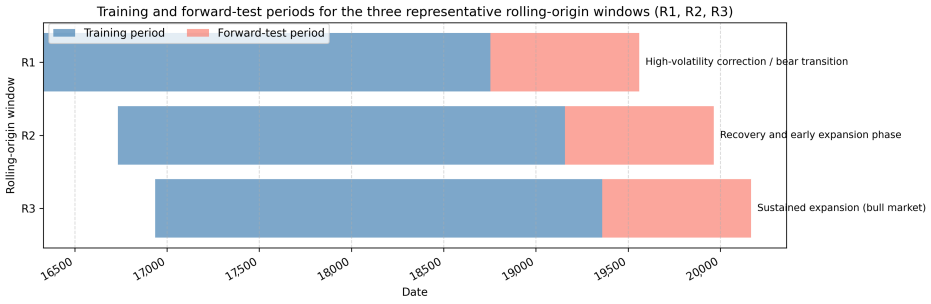

Figure 1. Timeline of training and forward-test periods for the three representative rolling-origin windows (R1, R2, R3). Blue bars represent training periods; salmon bars represent test periods. The dominant market character of each test window is shown on the right.

3.4. Rolling-Origin Trend Visualization

Figure 2 shows the temporal evolution of test-window mean and standard deviation during rolling-origin evaluation, illustrating the model's performance and stability across time.

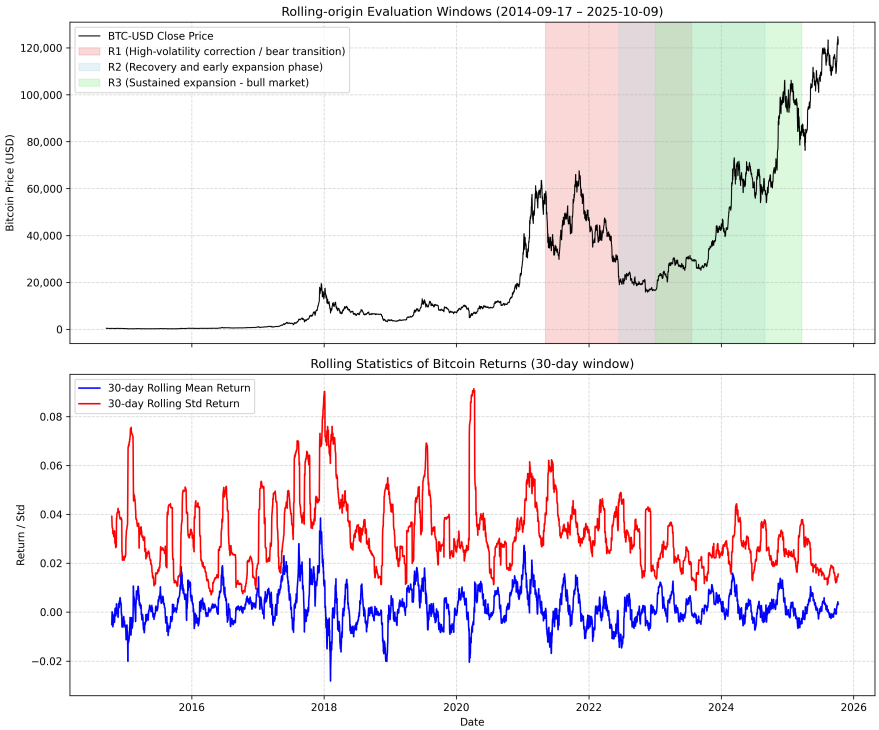

Figure 2. Rolling-origin evaluation of test-window mean and standard deviation across time (4040-day period, September 2014–October 2025; all images and figures were created by the authors of this paper).

3.5. Reproducibility and Methodological Justification

To ensure reproducibility and a fair comparison between the baseline and spiking models, all stochastic sources and evaluation conditions were explicitly controlled. The representative single-run results and diagnostic figures reported in the manuscript use a fixed random seed (`SEED = 25`), applied consistently across Python's `random`, NumPy, and the deep learning frameworks used in this study. For framework-level determinism, we set `tf.random.set_seed(SEED)` for the TensorFlow/Keras pipeline and `torch.manual_seed(SEED)` for the PyTorch pipeline; when using CUDA, deterministic execution settings were enabled (e.g., disabling cuDNN benchmarking) to reduce nondeterministic kernel selection. In addition, the statistical validation reported in Section 5.3 was conducted over 10 independent runs with different random seeds in order to assess performance stability beyond a single initialization.

Both models were trained and evaluated on the same expanded BTC–USD dataset described in Section 3.1, comprising 4039 daily closing-price observations from 17 September 2014 to 9 October 2025. An identical preprocessing pipeline was applied to both architectures, together with the same 30-day lookback window, the same strict chronological 80/20 hold-out split, and the same windowed regime-based robustness analysis, in order to prevent look-ahead bias and preserve comparability. Min–Max scaling to $[0, 1]$ was fitted on the training portion only and then applied to validation and test segments. The 30-day input window was retained after preliminary screening of candidate lookback lengths from 5 to 120 days, where performance stabilized around 30–35 days, while longer windows tended to increase overfitting under high volatility.

To match the optimization effort, in the hyperparameter search for both models, the same Grey Wolf Optimizer (GWO) budget (5 wolves $\times$ 5 iterations; 25 function evaluations) and the same two-stage training protocol were used. In Stage 1, each candidate configuration was trained for 5 epochs to obtain a fast and comparable validation-loss fitness estimate. In Stage 2, the best configuration was retrained up to 50 epochs with identical early stopping (patience = 5) and best-weight restoration. Across both models, AdamW was used with batch size 32 and 32 hidden units. The GWO was tuned only for the learning rate and weight decay; spiking-specific dynamics in GWO-SLSTM were learned during training via surrogate gradients.

Finally, all evaluation metrics (MSE, RMSE, MAPE, PBIAS) were computed on inverse-transformed USD prices using identical prediction–target pairs, and runtime was recorded under the same reporting protocol. Despite different implementation frameworks (TensorFlow/Keras for GWO-LSTM and PyTorch for GWO-SLSTM), using the controlled seeding strategy, identical data partitions, and matched training/evaluation procedures ensures reproducible experiments and a fair comparative analysis.

3.6. GWO-Optimized LSTM and SLSTM Architectures

We evaluate four forecasting configurations with matched recurrent capacity: a plain LSTM, a plain SLSTM, a Grey Wolf-Optimized LSTM (GWO-LSTM), and a Grey Wolf-Optimized Spiking LSTM (GWO-SLSTM). The plain models isolate architectural behavior without metaheuristic tuning, whereas the GWO variants isolate the contribution of hyperparameter optimization within each architecture. All four models use a single recurrent layer followed by a linear output head; the key architectural difference is that in SLSTM/GWO-SLSTM, the conventional hidden-state recurrence is replaced with LIF-inspired synaptic, membrane, and threshold dynamics, yielding a spiking-inspired recurrent mechanism based on thresholded event gating rather than hard-reset spike propagation.

3.6.1. Baseline LSTM

Given the input x_t, the LSTM updates gates, cell state, and hidden state as:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}), \tag{1}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}), \tag{2}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}), \tag{3}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}), \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{5}$$
$$h_t = o_t \odot \tanh(c_t). \tag{6}$$

3.6.2. Spiking LSTM (SLSTM)

The SLSTM preserves the LSTM gating structure but uses a synaptic state syn_t and membrane potential mem_t in place of (c_t, h_t):

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}\text{mem}_{t-1} + b_{hi}), \tag{7}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}\text{mem}_{t-1} + b_{hf}), \tag{8}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}\text{mem}_{t-1} + b_{hg}), \tag{9}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}\text{mem}_{t-1} + b_{ho}), \tag{10}$$
$$\text{syn}_t = f_t \odot \text{syn}_{t-1} + i_t \odot g_t, \tag{11}$$
$$\text{mem}_t = o_t \odot \tanh(\text{syn}_t). \tag{12}$$

A spike event is generated by thresholding the membrane potential:

$$S_t = \mathbb{I}(\text{mem}_t \geq V_{\text{th}}).$$

In a general spiking formulation, threshold crossing may optionally be followed by a hard reset, a subtractive reset, or no reset. In the reported experiments, the reset mechanism was fixed to `reset = 'none'`, meaning that when the membrane potential crosses the threshold, no hard or subtractive reset occurs. The membrane potential continues to evolve, allowing subthreshold information to accumulate across time steps. This design better preserves temporal dependencies in non-stationary financial series. The threshold V_{th} was learned during training (`learn_threshold = True`), allowing the effective firing condition to adapt to the data. Accordingly, in the present SLSTM, spiking is not a classical hard-reset event-driven mechanism; rather, the forecasting behavior is shaped by the interaction of thresholded event generation, synaptic and membrane-state dynamics, and adaptive threshold learning. Because S_t is non-differentiable, surrogate gradients are used in training to enable backpropagation through the spike-generation operation. The GWO is used to tune the conventional continuous training hyperparameters only (learning rate and weight decay) under a fixed evaluation budget.

3.6.3. Input Scenario and Fairness Controls

The main experiment features a strictly univariate input setting based on BTC–USD closing prices only, in order to isolate architectural effects and reduce feature-induced leakage risk. The plain LSTM and plain SLSTM have the same recurrent architectures as their GWO-optimized counterparts; the only difference is that the baseline variants are trained without GWO, using fixed training hyperparameters. This yields four directly comparable configurations: LSTM, SLSTM, GWO-LSTM, and GWO-SLSTM. The resulting pairwise comparisons isolate (i) the effect of spiking recurrence without metaheuristic tuning (LSTM vs. SLSTM), (ii) the effect of GWO within the conventional architecture (LSTM vs. GWO-LSTM), (iii) the effect of GWO within the spiking architecture (SLSTM vs. GWO-SLSTM), and (iv) the architecture effect under matched GWO budgets (GWO-LSTM vs. GWO-SLSTM). Figure 3 summarizes the executed pipeline only: univariate input, GWO tuning of learning rate and weight decay, and evaluation using MSE, RMSE, MAPE, and PBIAS.

For full transparency, Appendix B provides the language-agnostic pseudocode for the SLSTM forward pass, the GWO search loop, and the fitness evaluation routine, consistent with the hyperparameters in Table 3. In terms of asymptotic complexity, both LSTM and SLSTM forward passes scale linearly with sequence length and hidden dimension, while the overall training cost additionally scales with the fixed GWO evaluation budget, ensuring controlled and comparable computational overhead.

3.7. Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) is a population-based metaheuristic that emulates the leadership hierarchy and hunting strategies of grey wolves [23]. Solutions are ranked as α (best), β, δ, and ω, with searches guided by the top three to balance exploration and exploitation.

Search space and objective: The GWO tunes continuous hyperparameters for both models under identical budgets. Each wolf encodes $\mathbf{X}_i = [\eta_i, \text{WD}_i]$ (learning rate η, weight decay WD), with fitness as the validation MSE after 5 epochs (Stage 1, as per Section 3.5). Bounds are model-specific:

$$\eta \in [10^{-4}, 10^{-1}], \quad \text{WD} \in [10^{-4}, 9 \times 10^{-1}] \quad \text{(GWO-LSTM)},$$
$$\eta \in [10^{-4}, 10^{-1}], \quad \text{WD} \in [0, 10^{-3}] \quad \text{(GWO-SLSTM)}.$$

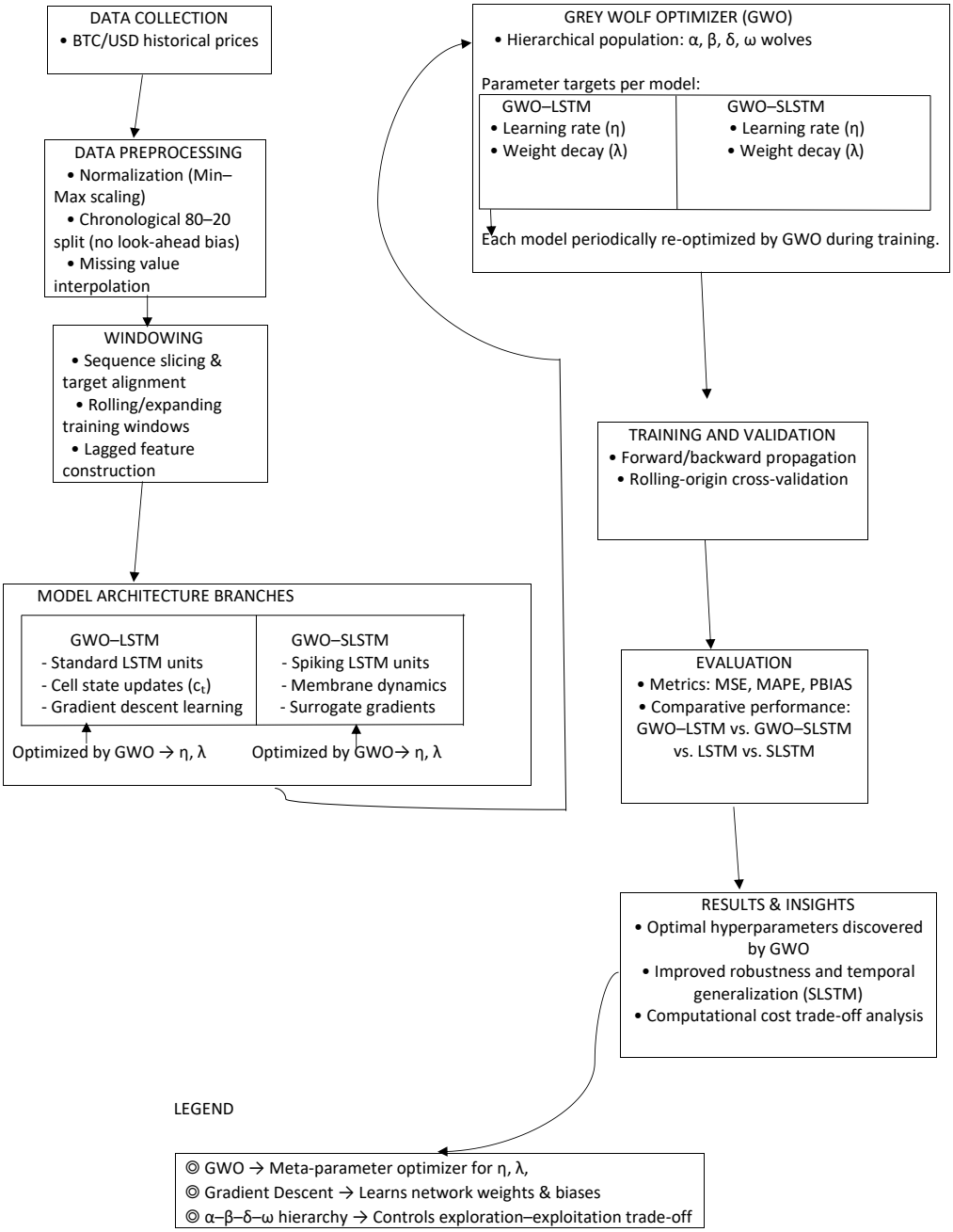

Figure 3. Corrected executed univariate forecasting pipeline used in the main experiments. BTC–USD closing prices are Min–Max normalized using training-set parameters, transformed into 30-day input windows, and evaluated under strict chronological hold-out testing. Four configurations are compared: LSTM, SLSTM, GWO-LSTM, and GWO-SLSTM. The GWO is applied only during the hyperparameter-search stage to tune the learning rate and weight decay under a fixed 25-evaluation budget; it is not periodically re-applied during final training. Model outputs are evaluated using MSE, RMSE, MAPE, and PBIAS. The SLSTM branch differs from LSTM through LIF-inspired synaptic, membrane, and adaptive-threshold dynamics.

Table 3. Hyperparameters for LSTM, GWO-LSTM, SLSTM, and GWO-SLSTM, including search bounds, fixed settings, and the final selected values used in the reported experiments.

Parameter	Type	Bounds/Choices	Final Value	Notes
Common (Both Models)				
Learning rate (η)	Continuous	$[10^{-4}, 10^{-1}]$	0.0052/0.0018	Optimized by GWO; AdamW
Weight decay (WD)	Continuous	$[10^{-4}, 9! \times !10^{-1}]$ (LSTM)/$[0, 10^{-3}]$ (SLSTM)	0.00012 (GWO-LSTM); 0.00079 (GWO-SLSTM)	Optimized by GWO; SLSTM upper bound 0.001
Batch size	Fixed	32	32	Held constant
Input window	Fixed	30 time steps	30	Held constant
Max training epochs	Fixed	50 (both)	50	Stage 2 only
Early stopping patience	Fixed	5 (both)	5	Stage 2 only
Architectural Differences				
Hidden units	Fixed	32 (LSTM), 32 (SLSTM)	32/32	Held constant; SLSTM uses `hidden_size = 32` in `BitcoinPredictor`
Dropout	Fixed	0.0 (both)	0.0/N/A	No dropout used in either model
SLSTM-Specific (Spiking Dynamics)				
Spike threshold (V_{th})	Learned	Adaptive during training	Learned	`learn_threshold = True`
Reset mechanism	Fixed	`'none'`	`'none'`	No hard or subtractive reset applied after threshold crossing; membrane state continues across time steps
Surrogate gradient	Fixed	`surrogate.atan()`	`atan`	Arctan surrogate for spike function
LSTM-Specific				
Activation function	Fixed	tanh (recurrent), sigmoid (gates)	Standard LSTM	Default TensorFlow/Keras
Recurrent dropout	Fixed	0.0	0.0	No recurrent dropout
GWO Configuration (LSTM)				
Population size	Fixed	5	5	25 evaluations total
Iterations	Fixed	5	5	
GWO Configuration (SLSTM)				
Population size	Fixed	5	5	25 evaluations total (as per source code)
Iterations	Fixed	5	5	
GWO Common Settings				
Fitness epochs	Fixed	5	5	Stage 1 only
Search dimension	Fixed	2 (η, WD)	2	Continuous variables
Random seed	Fixed	25	25	Reproducibility

The best configurations are used in full training (Stage 2). Recent variants like attention-based GWO (AtGWO) [38] and hybrid GWO-based neural optimization frameworks [39,40] further improve convergence and hyperparameter tuning robustness for neural networks, motivating our adoption.

Position update: The convergence factor decreases linearly:

$$a(t) = 2 - 2\frac{t}{T_{\max}}. \tag{13}$$

For leaders $\ell \in \{\alpha, \beta, \delta\}$, with $\mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{U}(0,1)^D$:

$$\mathbf{A}_\ell = a(2\mathbf{r}_1 - 1), \quad \mathbf{C}_\ell = 2\mathbf{r}_2, \tag{14}$$

$$\mathbf{D}_\ell = |\mathbf{C}_\ell \odot \mathbf{X}_\ell - \mathbf{X}_i|, \tag{15}$$

$$\mathbf{X}'_\ell = \mathbf{X}_\ell - \mathbf{A}_\ell \odot \mathbf{D}_\ell, \tag{16}$$

$$\mathbf{X}_i^{\text{new}} = \frac{1}{3}(\mathbf{X}'_\alpha + \mathbf{X}'_\beta + \mathbf{X}'_\delta). \tag{17}$$

Positions are clipped to bounds; loop runs for $T_{\max} = 5$ iterations with five wolves (25 evaluations).

Integration with gradient-based learning: GWO optimizes only η and WD (low-dimensional, per Table 3); the spiking parameters in GWO-SLSTM use surrogate gradients. This hybridization enables robust generalization in volatile series, as demonstrated in GWO-based neural network optimization frameworks [39].

The GWO configuration implemented in this study is the standard low-budget setup described above, and was used consistently across all evaluated models.

3.8. Training and Evaluation Protocol

The GWO-optimized models were trained using the AdamW optimizer with early stopping based on the validation MSE. A patience of five epochs was used to reduce overfitting during final training. After the best hyperparameter configuration was identified in Stage 1 from the validation results, that configuration was retrained in Stage 2 on the combined training and validation data for up to 50 epochs, again with early stopping (patience = 5) and best-weight restoration. The baseline LSTM and SLSTM models were trained under the same data splits and stopping logic, but without a GWO-based hyperparameter search.

The predictive performance was assessed using mean squared error (MSE), root mean squared error (RMSE), mean absolute percentage error (MAPE), and percent bias (PBIAS). All metrics were computed on inverse-transformed BTC–USD prices using identical prediction–target pairs, enabling direct comparison across the four forecasting configurations. In addition, training and testing times were recorded to evaluate computational efficiency alongside predictive accuracy.

4. Experimental Setup

All experiments were executed on a high-performance server equipped with an NVIDIA Tesla V100 GPU, an Intel Xeon Platinum CPU, and 64 GB of RAM. the softwars used in this experiment are numpy: 1.24.3, torch: 2.2.2, tensorflow: 2.16.2, pandas: 2.3.1, matplotlib: 3.10.0, sklearn: 1.2.2, snntorch: 0.9.4, tqdm: 4.67.1, statsmodels: 0.14.5, seaborn: 0.13.2, yfinance: 0.2.66,

Implementation frameworks:

- **LSTM/GWO-LSTM:** Implemented in TensorFlow/Keras, leveraging highly optimized LSTM primitives. The plain LSTM serves as the non-optimized conventional baseline,

whereas GWO-LSTM has the same architecture with the learning rate and weight decay selected by the GWO.

- **SLSTM/GWO-SLSTM:** Implemented in PyTorch 2.2.2 with `snnTorch`, enabling explicit implementation of spiking neuron dynamics and surrogate-gradient training. The plain SLSTM serves as the non-optimized spiking baseline, whereas GWO-SLSTM uses the same architecture with learning rate and weight decay selected by GWO.

The framework choice reflects implementation requirements rather than differences in experimental conditions: TensorFlow/Keras provides stable, production-grade recurrent layers, while PyTorch offers the flexibility needed to define custom spiking-state updates. To preserve comparability, both pipelines used the same data partitions and preprocessing, identical random seeds, and matched training/evaluation procedures. All runs were conducted in a Python 3.9 environment with deterministic settings where supported, and the hardware/software configuration was kept constant across experiments to ensure a fair comparison between conventional and spiking architectures.

4.1. Hyperparameter Search Configuration

Hyperparameter optimization for both GWO-LSTM and GWO-SLSTM was performed using the Grey Wolf Optimizer. The search was restricted to two continuous hyperparameters only: the learning rate and the weight decay. All remaining parameters were held fixed across optimization runs to preserve comparability between architectures. In particular, spiking-specific settings such as the surrogate gradient, reset mode, and threshold-learning mechanism were not optimized by GWO; they were fixed by design or learned through gradient-based training within the SLSTM model. The optimization followed a two-stage procedure. In Stage 1 (fast evaluation), each candidate configuration was trained for 5 epochs, and the resulting validation MSE was used as the fitness score. In Stage 2 (final training), the best configuration from Stage 1 was retrained using the full training protocol with early stopping and best-weight restoration.

The learning-rate search range was the same for both architectures,

$$\eta \in [10^{-4}, 10^{-1}],$$

while the weight-decay range was architecture-specific:

$$\mathrm{WD} \in [10^{-4}, 9 \times 10^{-1}] \quad \text{for GWO-LSTM,}$$

$$\mathrm{WD} \in [0, 10^{-3}] \quad \text{for GWO-SLSTM.}$$

The GWO search budget was matched across both architectures at 25 candidate evaluations, corresponding to five wolves and five iterations for each model. This matched-budget design is important because it ensures that differences between GWO-LSTM and GWO-SLSTM are not attributable to unequal optimization effort. The complete set of fixed settings, search bounds, and final selected values is summarized in Table 3.

Both model families used the same number of fitness-evaluation epochs (five epochs per candidate) and the same early-stopping patience (5). For the SLSTM variants, the reset mode was fixed to `'none'` and the firing threshold was learned during training. Thus, threshold crossings generated spike events but did not trigger either a hard reset to zero or a subtractive reset of the membrane potential. This choice was deliberate: for financial forecasting, retaining subthreshold information across time steps may be more useful than enforcing temporally sparse resets. The adaptive threshold helped stabilize membrane dynamics under this no-reset configuration. We therefore describe the reported SLSTM as

a spiking-inspired recurrent model with thresholded event gating, rather than as a classical hard-reset spiking network.

4.2. Evaluation Metrics and Statistical Testing

Model accuracy was assessed using four standard regression measures: mean squared error (MSE), which emphasizes larger absolute errors; root mean squared error (RMSE), which reports the error magnitude for the original price units; mean absolute percentage error (MAPE), which reflects relative error; and percent bias (PBIAS), which captures systematic directional error. Metrics were computed on inverse-transformed prices (USD) using identical prediction–target pairs for both the training and test splits.

PBIAS captures systematic directional error:

$$\text{PBIAS} = 100 \times \frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)}{\sum_{t=1}^{N} y_t}.$$

A positive PBIAS indicates underestimation ($\hat{y}_t < y_t$ on average), while negative values indicate overestimation. Unlike MAPE, which reflects error magnitude, PBIAS highlights directional drift and therefore complements magnitude-based metrics in volatile, non-stationary series [41–46].

To compare models, we performed paired t-tests over 10 independent runs with different random seeds. In addition to p-values, relative percentage improvements are reported to provide an interpretable measure of practical effect size beyond statistical significance.

5. Results and Analysis

This section reports the empirical findings of the study. It first examines forecast behavior and rolling-error patterns, then presents training dynamics, comparative benchmark results, and computational-efficiency observations for the four evaluated model configurations.

5.1. Target Prediction Analysis

Figures 4 and 5 compare the actual BTC–USD test trajectory with the LSTM and GWO-LSTM predictions, showing that GWO-LSTM tracks the series more closely (correlation = 0.998) than LSTM (correlation = 0.990), although both follow the overall trend.

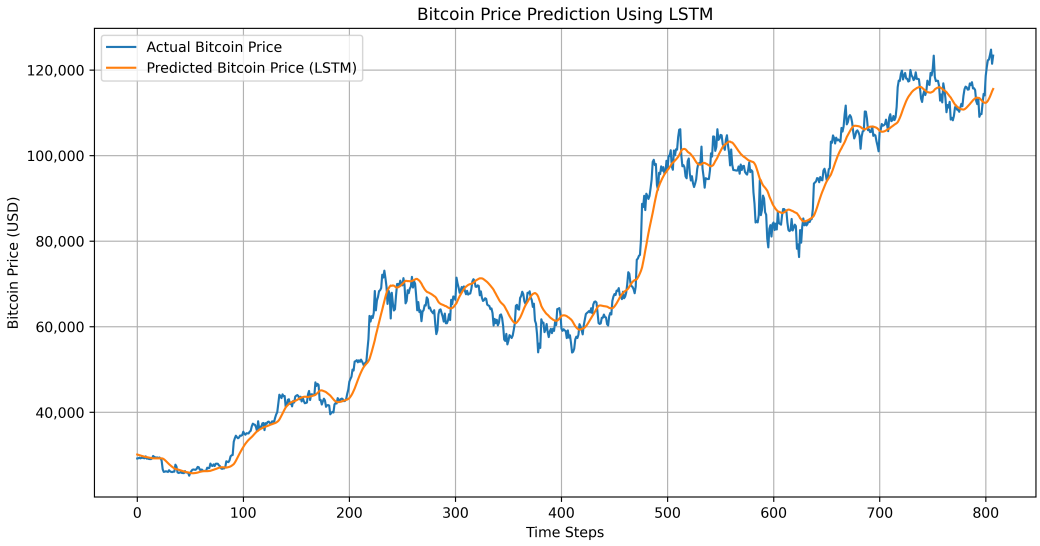

Figure 4. Actual vs. predicted BTC prices using standard LSTM. The model captures the overall trend (correlation = 0.990) but shows a slight systematic bias (mean actual $72,108.64 vs. predicted $72,083.94).

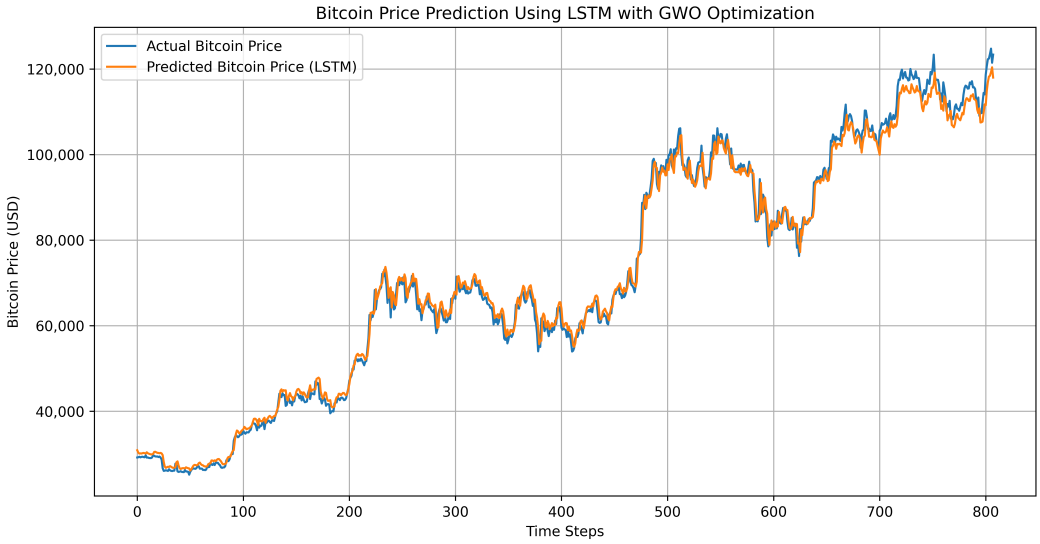

Figure 5. Actual vs. predicted BTC prices using GWO-LSTM. The model achieves stronger agreement (correlation = 0.998) with a mean predicted price of $72,030.90.

Figures 6 and 7 show the prediction performance of SLSTM and GWO-SLSTM, respectively. The SLSTM model (Figure 6) yields a correlation coefficient of 0.993 with an average predicted value of $71,290.94, while the GWO-optimized SLSTM (Figure 7) achieves the highest correlation (0.998) and a mean predicted price of $71,852.06, which is closer to the actual average of $72,108.64 compared to the non-optimized SLSTM.

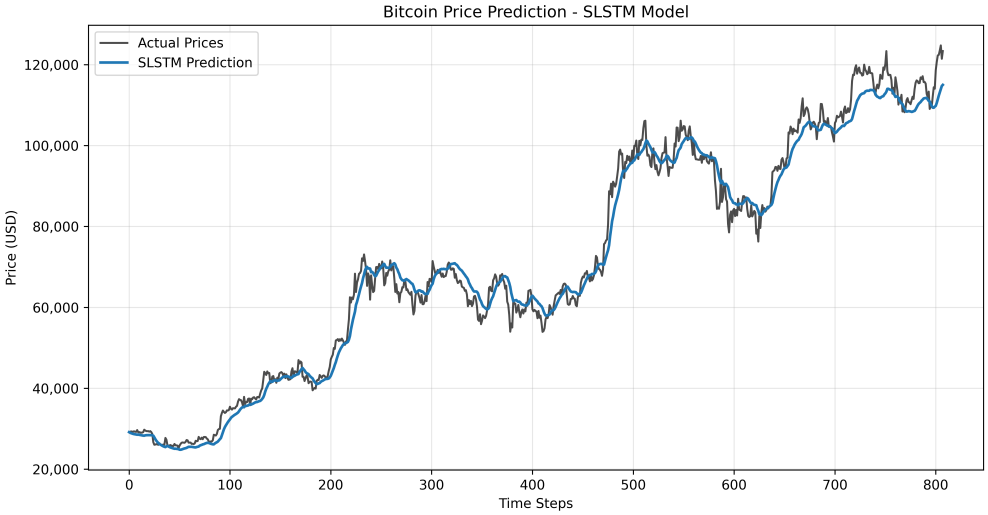

Figure 6. Actual vs. predicted BTC prices using SLSTM. The model shows good agreement (correlation = 0.993) with a mean predicted price of $71,290.94.

The above performance graphs are complemented with Seven-day rolling RMSE and MAPE for all models. See Appendix B for the seven-day rolling analysis. In the main text below we are only showing below the Seven-day rolling RMSE and MAPE for GWO-SLSTM in Figure 8 because it achieves the lowest rolling RMSE and MAPE among the evaluated configurations.

5.2. Training Dynamics

Figures 9–12 depict the convergence pattern when the same early stopping rule is employed (patience = 5). While LSTM (Figure 9) reached a validation plateau very quickly and stopped training after only 5 epochs, ending with a large final validation MSE

(32, 543, 626.22), GWO-LSTM (Figure 10) continues converging for 26 epochs, achieving a much smaller validation MSE (6, 278, 161.21).

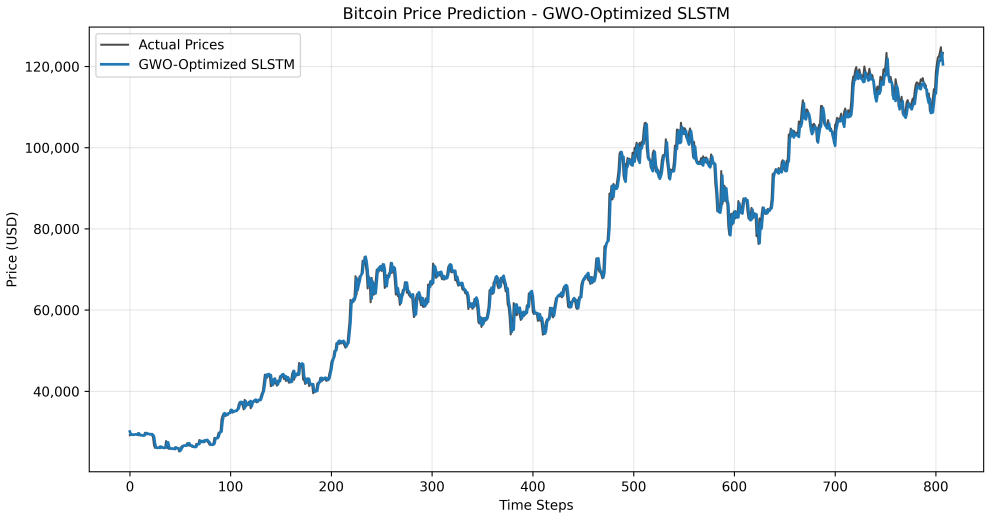

Figure 7. Actual vs. predicted BTC prices using GWO-SLSTM. The model achieves the strongest agreement (correlation = 0.998) with a mean predicted price of $71,852.06, bringing it closer to the actual mean of $72,108.64 than the standard SLSTM.

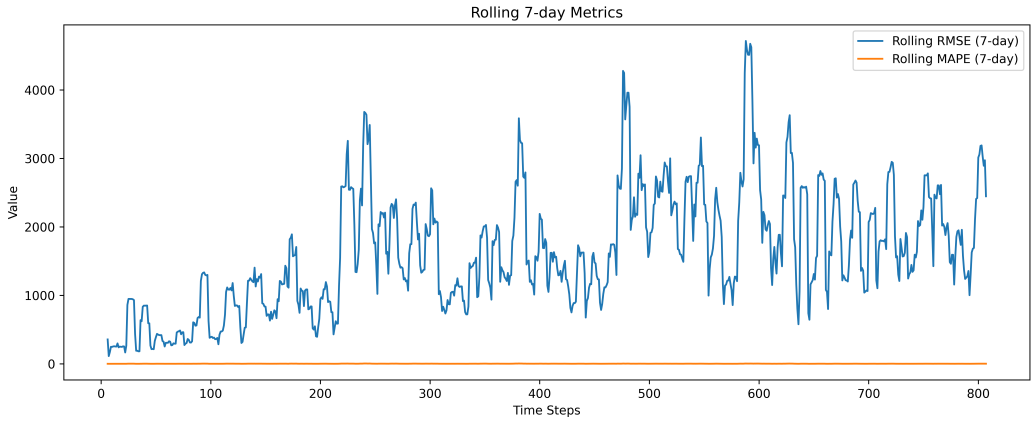

Figure 8. Seven-day rolling RMSE and MAPE for GWO-SLSTM. This model achieves the lowest rolling RMSE and MAPE among the evaluated configurations (average RMSE = 1624.47, max = 4715.67; average MAPE = 1.76%, max = 5.13%), indicating comparatively lower rolling errors under the evaluated regime-shifting conditions.

Unlike LSTM, SLSTM (Figure 11) converges after 6 epochs, producing a validation MSE of 29, 990, 066. However, GWO-SLSTM exhibits the most notable optimization process (Figure 12), converging rapidly and stopping after 20 epochs with the best validation MSE (4, 928, 175) and the lowest validation loss among all models.

5.3. Benchmarking Results

Table 4 reports the representative single-run chronological hold-out experiment using `SEED = 25`. A repeated-run statistical comparison over 10 independent runs is then used to assess the stability of these findings beyond seed-specific behavior. In the representative hold-out experiment, GWO-SLSTM reduced the test RMSE from 3501.48 to 1840.97 relative to the baseline SLSTM, reduced the test MAPE from 3.86% to 1.76%, and maintained a near-zero PBIAS of 0.36%.

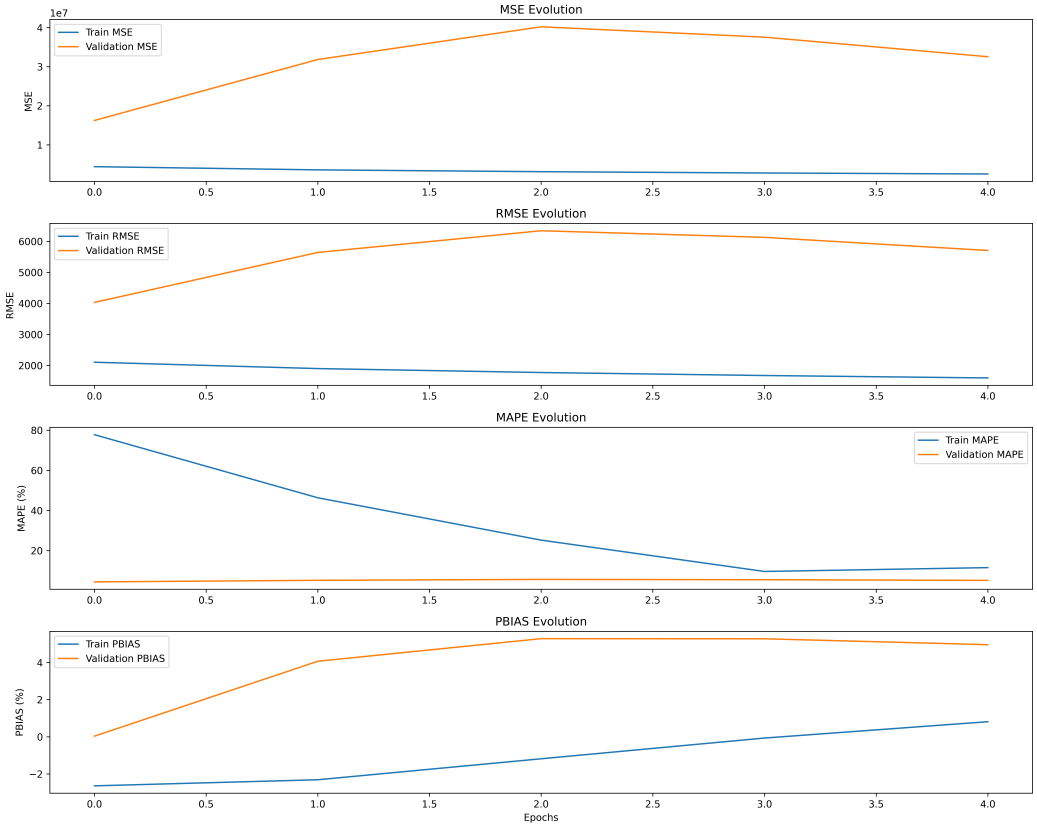

Figure 9. Training and validation curves for standard LSTM. Early stopping triggers at epoch 5 with high validation error, indicating rapid overfitting or convergence to a suboptimal minimum.

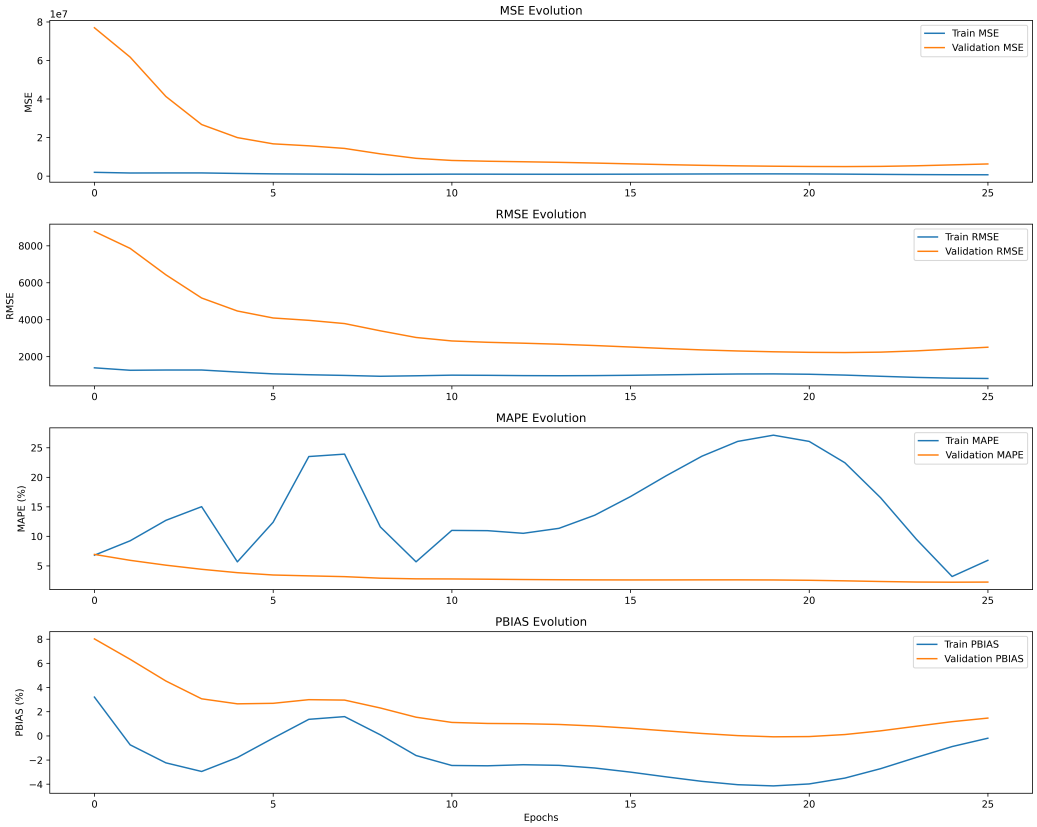

Figure 10. Training and validation curves for GWO-LSTM. The model continues to improve for 26 epochs, achieving a significantly lower validation error than standard LSTM.

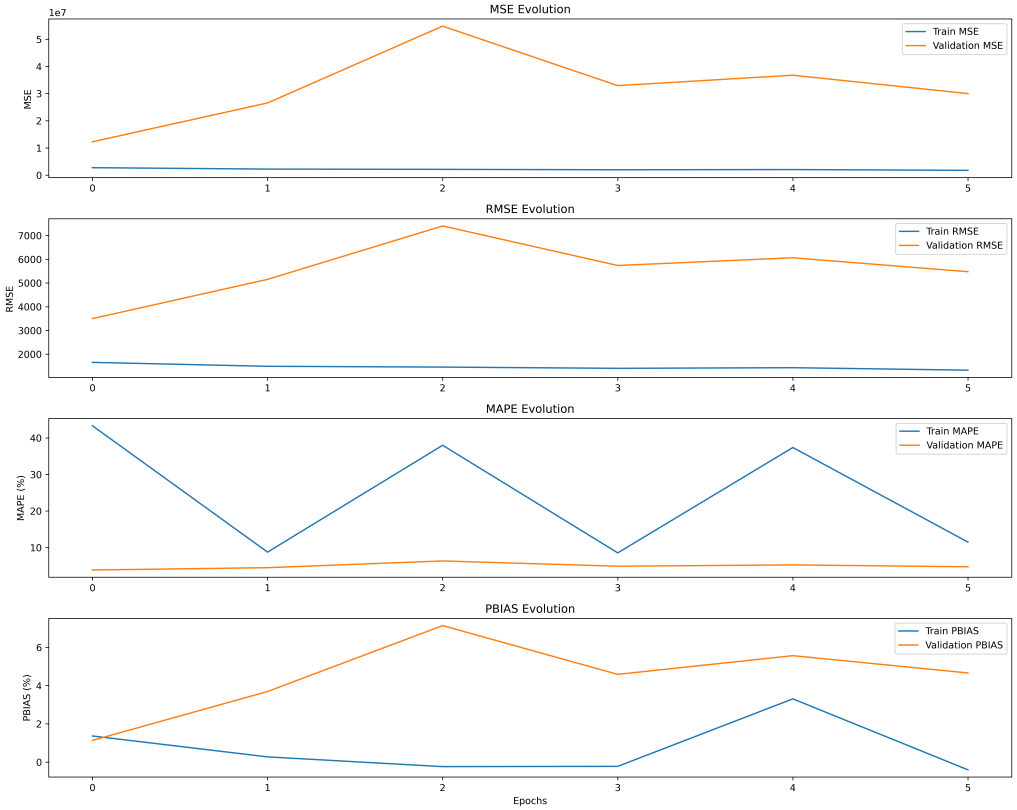

Figure 11. Training and validation curves for standard SLSTM. Early stopping triggers at epoch 6, showing improved but not optimal stability.

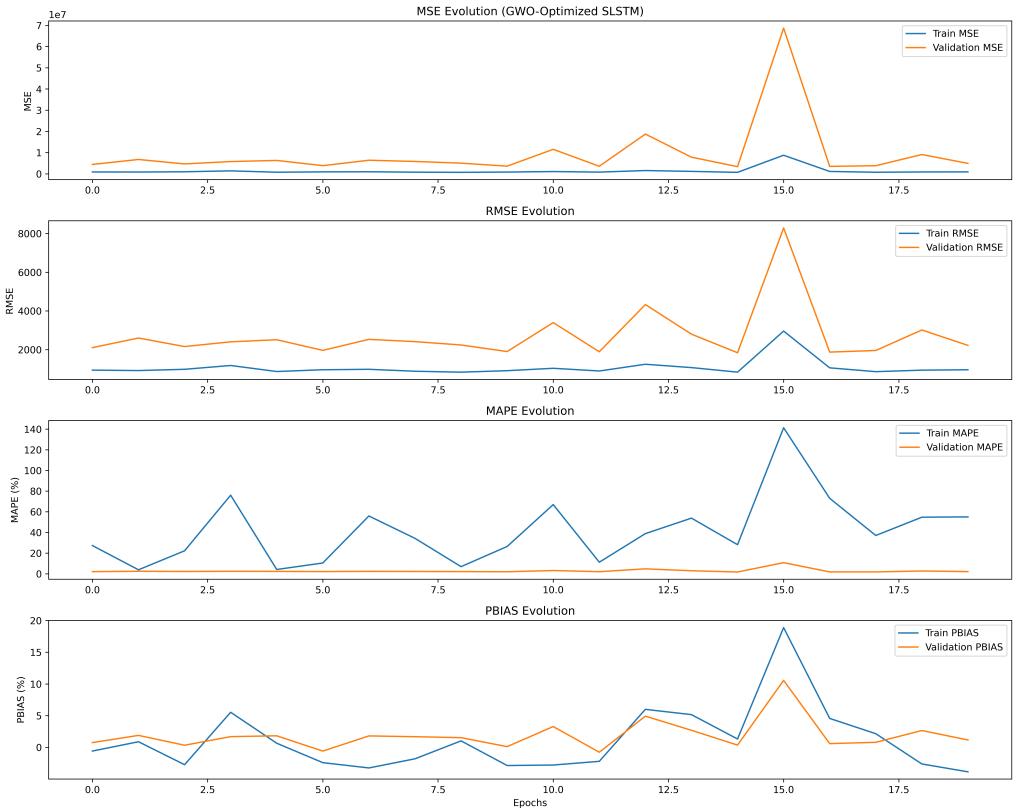

Figure 12. Training and validation curves for GWO-SLSTM. The model achieves the best validation loss and stable convergence, suggesting that the combination of spiking-inspired recurrence and GWO optimization was associated with improved convergence behavior under the evaluated conditions.

The quantitative window-specific robustness results are reported directly in Table 5 . Because the final stored predictions cover the primary hold-out prediction horizon, R1 lies outside the available scored prediction period and is therefore reported as not evaluated. R2 is partially covered and R3 is largely covered. Across the two evaluable windows, GWO-SLSTM achieved the lowest RMSE and MAPE, followed by GWO-LSTM, SLSTM, and LSTM. This provides quantitative evidence that the main hold-out ranking is preserved within the available windowed robustness analysis, while avoiding the overstatement that all three rolling-origin windows were fully retrained and scored.

The four-model comparison was designed to determine where the observed gains originate. The LSTM–SLSTM comparison isolates the architectural contribution of the spiking-inspired recurrence without metaheuristic tuning, while the LSTM–GWO-LSTM and SLSTM–GWO-SLSTM comparisons isolate the contribution of GWO within each architecture. Lastly, the GWO-LSTM–GWO-SLSTM comparison evaluates the architecture effect under matched search budgets and evaluation conditions. These four configurations are summarized in Table 4 under the same preprocessing, data partitions, and evaluation metrics.

Table 4. Performance comparison across LSTM variants. Values are in USD, except MAPE and PBIAS (%). GWO-LSTM and GWO-SLSTM results are from optimized configurations; baseline LSTM and SLSTM use the fixed Adam optimizer (lr = 0.001, weight decay = 0).

Metric	Baseline (Adam)		GWO-Optimized	
	LSTM	**SLSTM**	**GWO-LSTM**	**GWO-SLSTM**
		Training		
RMSE	2103.08	1656.66	996.15	839.49
MSE	4,422,931.52	2,744,508.00	992,312.79	704,744.13
MAPE (%)	77.79	43.35	22.46	28.18
PBIAS (%)	−2.64	1.37	−3.50	1.32
		Test		
RMSE	4030.10	3501.48	2217.24	1840.97
MSE	16,241,684.70	12,260,338.00	4,916,169.91	3,389,182.50
MAPE (%)	4.40	3.86	2.46	1.76
PBIAS (%)	0.03	1.13	0.11	0.36
		Training Configuration		
Early stopping	Epoch 5	Epoch 6	Epoch 26	Epoch 20
Best val. loss	0.001047	—	0.000317	—

The high training MAPE values relative to testing are due to the extreme non-stationarity and low absolute prices in the early years of the Bitcoin dataset. MAPE is sensitive to low-magnitude values, which inflate percentage errors in the training period. Therefore, MAPE should be interpreted cautiously and is less informative than RMSE for non-stationary cryptocurrency forecasting.

5.3.1. Windowed Robustness Analysis Setup

To present the regime-specific evidence more quantitatively in the main manuscript, Table 5 reports the window-specific model performance for the portions of the pre-specified forward-test windows that overlap with the available prediction horizon. Metrics were computed on inverse-transformed BTC–USD prices. Models are ranked within each evaluable window by RMSE, where a lower rank indicates better predictive performance.

Table 5. Window-specific robustness results across the pre-specified forward-test regimes. R1 is shown for transparency but is not scored because it does not overlap with the stored prediction horizon. Quantitative model comparisons are therefore based on the evaluable portions of R2 and R3.

Window	Forward-Test Period	Evaluated Sub-Period	Model	RMSE	MAPE (%)	PBIAS (%)	Rank
R1	7 May 2021–23 July 2023	Not covered by stored predictions	LSTM	–	–	–	–
			SLSTM	–	–	–	–
			GWO-LSTM	–	–	–	–
			GWO-SLSTM	–	–	–	–
R2	15 June 2022–30 August 2024	24 July 2023–30 August 2024	LSTM	3456.57	4.93	−0.72	4
			SLSTM	2790.86	4.36	0.33	3
			GWO-LSTM	1711.39	2.78	−1.86	2
			GWO-SLSTM	1438.62	1.80	−0.13	1
R3	3 January 2023–20 March 2025	24 July 2023–20 March 2025	LSTM	4008.73	4.84	−0.73	4
			SLSTM	3148.13	4.07	0.25	3
			GWO-LSTM	1936.34	2.56	−0.92	2
			GWO-SLSTM	1760.52	1.86	0.08	1

5.3.2. Window-Specific Interpretation

The window-specific results in Table 5 show a consistent ranking across the two evaluable windows. In R2, GWO-SLSTM achieved the lowest RMSE (1438.62) and MAPE (1.80%), followed by GWO-LSTM (RMSE = 1711.39; MAPE = 2.78%), SLSTM (RMSE = 2790.86; MAPE = 4.36%), and LSTM (RMSE = 3456.57; MAPE = 4.93%). In R3, the same ordering was observed: GWO-SLSTM again ranked first (RMSE = 1760.52; MAPE = 1.86%), followed by GWO-LSTM (RMSE = 1936.34; MAPE = 2.56%), SLSTM (RMSE = 3148.13; MAPE = 4.07%), and LSTM (RMSE = 4008.73; MAPE = 4.84%).

Averaged across R2 and R3, GWO-SLSTM reduced RMSE by approximately 57.1% relative to LSTM, 46.1% relative to SLSTM, and 12.3% relative to GWO-LSTM. These results provide direct quantitative support in the main text for the claim that the optimized spiking-inspired recurrent configuration was the strongest-performing model within the evaluated windowed robustness analysis. At the same time, because R1 was not scored and the models were not fully retrained independently for every rolling origin, these results should be interpreted as a regime-window robustness check over the available prediction horizon rather than as a complete rolling-origin re-estimation benchmark.

To investigate whether the improvement holds across multiple independent runs, we performed paired *t*-tests across 10 independent runs with different random seeds. The difference between GWO-LSTM and conventional LSTM and between GWO-SLSTM and SLSTM was found to be statistically significant, with a reduction in RMSE of $p < 0.001$. The repeated-run evaluation further supports the stability of the observed improvement. Representative point estimates from the SEED = 25 trial are:

- Test RMSE: 1840.97 (GWO-SLSTM) vs. 3501.48 (baseline SLSTM);
- Test RMSE: 2217.24 (GWO-LSTM) vs. 4030.10 (baseline LSTM);
- Test MAPE: 1.76% (GWO-SLSTM) vs. 3.86% (baseline SLSTM);
- Test MAPE: 2.46% (GWO-LSTM) vs. 4.40% (baseline LSTM);
- Test PBIAS: 0.36% (GWO-SLSTM) vs. 1.13% (baseline SLSTM).

GWO-based optimization significantly improved both LSTM and SLSTM architectures. LSTM exhibited a large difference between training and testing errors (RMSE: 2103 vs. 4030), suggesting overfitting or problems related to non-stationary dynamics. The proposed GWO-LSTM successfully reduced the difference in errors while also providing better results in terms of absolute test errors (RMSE: 2217). In addition, the spiking-inspired architecture exhibited lower test errors than the conventional LSTM baseline. The baseline SLSTM exhibited a better test RMSE compared to LSTM, with values of 3501 and 4030,

respectively. The optimized GWO-SLSTM further improved the test RMSE, achieving the best results among all methods (1841). Additionally, GWO-SLSTM exhibits the lowest test PBIAS value (0.36%), approaching zero. Notably, while GWO-LSTM achieves a slightly higher test MAPE (2.46%) than GWO-SLSTM (1.76%), the latter demonstrates superior RMSE and bias characteristics, indicating a more balanced error distribution across the test set.

5.3.3. Visual Representations

Additional diagnostic visualizations (radar charts, Taylor diagrams, and APE boxplots) were generated to cross-check agreement, variance matching, and error dispersion. These plots and their descriptions are provided in the Supplementary Material and are available at the repository link reported in the Data Availability Statement.

5.3.4. Computational Efficiency

To complement the accuracy results, we evaluated computational efficiency by recording end-to-end wall-clock time for training and testing under the same hardware and reporting protocol. Table 6 summarizes the timing metrics for all four model variants.

Table 6. Computational efficiency comparison across LSTM variants. All measurements are wall-clock seconds under identical hardware conditions.

Metric	LSTM	GWO-LSTM	SLSTM	GWO-SLSTM
Training time (s)	7.75	137.28	8.42	39.71
Testing time (s)	0.37	0.38	0.02	0.02
Total time (s)	8.13	137.66	8.44	39.73
Train/test time ratio	20.89	363.65	453.09	1610.66
Number of epochs	5	26	6	20
Train time per sample per epoch (s)	0.00048	0.00165	0.00044	0.00062
Test time per sample (s)	0.00046	0.00047	0.00002	0.00003
Early stopping	Epoch 5	Epoch 27	Epoch 6	Epoch 20
Best validation loss	0.001047	0.000317	—	0.000292

Several key observations emerge from the timing analysis. First, the baseline LSTM (Adam optimizer, no GWO) is the fastest model overall, completing training in just 7.75 s over five epochs before early stopping triggers. However, this speed comes at the cost of a poor predictive accuracy (test RMSE = 4030.10) and indicates rapid convergence to a suboptimal minimum.

The GWO-LSTM model requires substantially more computational resources (137.28 s training time, 26 epochs) due to the hyperparameter optimization search and the computational overhead of conventional LSTM layers. Its train time per sample per epoch (0.00165 s) is approximately 3.4× higher than the baseline LSTM.

The SLSTM architecture demonstrates excellent efficiency, requiring only 8.42 s of training time—slightly less than even the baseline LSTM—while achieving significantly better test accuracy (test RMSE = 3501.48). The spiking design yields a very low per-epoch cost (0.00044 s per sample per epoch), which is actually lower than that of the baseline LSTM (0.00048 s). Within the present implementation, the SLSTM exhibited a comparatively low per-epoch computational cost.

In the present implementation, the GWO-SLSTM model achieved the best predictive accuracy (test RMSE = 1840.97, test MAPE = 1.76%) while incurring a moderate training time of 39.71 s over 20 epochs; this runtime should be interpreted as setup-specific rather than as evidence of intrinsic architectural efficiency. Its train time per sample per epoch

(0.00062 s) is about 1.4× higher than the non-optimized SLSTM, but still far lower than the GWO-LSTM's per-epoch cost.

Important caveats and interpretation. The reported runtime differences should be interpreted in light of several important limitations. First, the two model families were implemented in different deep learning frameworks: TensorFlow/Keras for the LSTM variants (LSTM and GWO-LSTM) and PyTorch/snnTorch for the SLSTM variants (SLSTM and GWO-SLSTM). Framework-level differences in computational graph optimization, memory management, and operator execution can substantially influence wall-clock measurements independent of architectural efficiency. Second, the models stopped after different numbers of epochs due to early stopping (5, 26, 6, and 20 epochs, respectively), meaning that the GWO-LSTM's longer training time partly reflects its longer optimization path rather than solely its per-step computational cost. Third, the GWO search budgets were matched in terms of candidate evaluations (5×5), but the internal cost per candidate differs between frameworks.

Given these confounding factors, the observed speed advantage of GWO-SLSTM over GWO-LSTM (39.71 s vs. 137.28 s) should not be interpreted as evidence that the spiking architecture is intrinsically more computationally efficient than conventional LSTM. Rather, these measurements reflect the specific implementation choices, framework characteristics, and optimization trajectories used in this study. The key practical finding is that, within our implemented environment, GWO-SLSTM achieved the lowest RMSE and MAPE among the four evaluated configurations while incurring a moderate and practically acceptable training cost (under 40 s).

For real-time inference, however, the testing time per sample for both SLSTM (0.00002 s) and GWO-SLSTM (0.00003 s) is an order of magnitude smaller than the LSTM-based models. The observed inference-time advantage in this implementation is practically relevant, but it should still be interpreted cautiously because framework-level effects were not fully controlled. Thus, while training-time comparisons must be made cautiously, the inference efficiency of spiking architectures remains a practically relevant benefit for deployment scenarios requiring rapid, low-latency predictions.

In summary, within the reported software/hardware setup, the baseline LSTM is fastest but least accurate, while GWO-SLSTM achieves the best accuracy at a moderate training cost—a reasonable trade-off for offline training. However, the runtime differences should be understood as empirical measurements specific to this implementation, not as generalizable claims of architectural superiority.

6. Discussion

In this section, the empirical results are interpreted in relation to the study objectives and the broader literature. The main findings are summarized, the relative roles of spiking recurrence and GWO-based optimization are discussed, and then the principal limitations and directions for future work are outlined.

6.1. Summary of Key Findings

This study compared a Grey Wolf-Optimized Spiking LSTM (GWO-SLSTM) with three controlled counterparts—plain LSTM, plain SLSTM, and GWO-LSTM—to isolate the contributions of spiking recurrence and GWO-based hyperparameter optimization. For daily BTC–USD prices (17 September 2014–9 October 2025), GWO-SLSTM achieved a test RMSE of 1840.97 versus 2217.24 for GWO-LSTM (a 17.0% reduction) and a near-zero bias (PBIAS = 0.36% vs. 0.11% for GWO-LSTM). The test MAPE of GWO-SLSTM was 1.76%, which is also lower than the 2.46% for GWO-LSTM. This indicates that, within the present study, the optimized spiking architecture achieved both better absolute error

control and better proportional accuracy than the optimized conventional baseline. The total training time for GWO-SLSTM was 39.71 s, compared to 137.28 s for GWO-LSTM; however, this runtime difference should be interpreted as setup-specific rather than as evidence of intrinsic architectural efficiency.

In interpreting these gains, we note that the reported SLSTM uses thresholded event generation with `reset = 'none'`, so the architectural distinction from conventional LSTM arises from spiking-inspired recurrent dynamics and adaptive thresholding rather than from hard-reset spike events alone.

6.2. Complementary Error Characteristics and Architectural Incomparability

A central purpose of the revised comparative design is attribution. The plain LSTM–SLSTM comparison indicates the architectural contribution of the spiking recurrence before metaheuristic tuning: the baseline SLSTM improves the test RMSE from 4030.10 to 3501.48 relative to the baseline LSTM. The within-architecture comparisons then quantify the optimization effect of GWO: the test RMSE decreases from 4030.10 to 2217.24 for the conventional architecture (LSTM vs. GWO-LSTM) and from 3501.48 to 1840.97 for the spiking architecture (SLSTM vs. GWO-SLSTM). Finally, the GWO-LSTM vs. GWO-SLSTM comparison shows that, under matched GWO budgets and identical evaluation conditions, the optimized spiking-inspired architecture yielded the lowest error among the four evaluated configurations in this study.

Due to fundamental structural differences (continuous vs. discrete dynamics, surrogate gradients vs. standard BPTT, and different implementation frameworks), we do not claim general cross-architectural superiority. Rather, we interpret the observed gains as arising from both sources in this experimental setting: the spiking-inspired recurrence was associated with lower test error than the plain LSTM baseline, and GWO further improved both architecture families.

6.3. Interpretation and Practical Implications

The observed performance gains of GWO-SLSTM in this study are consistent with a beneficial interaction between spiking-inspired recurrence and evolutionary hyperparameter selection under the present evaluation protocol. In the present experiments, the spiking-inspired recurrence was associated with lower RMSE and MAPE under regime-shifting conditions. The GWO provides a low-budget, population-based search that improves convergence reliability without manual tuning, and it proved equally effective for the spiking architecture. Notably, the training time of GWO-SLSTM (39.71 s) was less than one-third of that of GWO-LSTM (137.28 s), while testing times were similarly negligible ($\approx$0.02 s total). Within the present implementation and evaluation protocol, this result suggests that the spiking variant need not be more computationally demanding, although the comparison remains setup-specific.

Practical guidance:

- **GWO-LSTM:** Suitable when the user is already invested in the TensorFlow/Keras ecosystem and requires a simple, drop-in optimization. Its training cost is higher than GWO-SLSTM, so it is not recommended unless framework constraints apply.
- **GWO-SLSTM:** The best-performing configuration within the present four-model comparison: it achieved the lowest RMSE and MAPE, maintained near-zero bias, trained in under 40 s in the present setup, and tested in milliseconds. It appears promising for volatile financial time series, while deployment on neuromorphic or edge hardware remains a direction for future validation rather than a demonstrated result of this study.

6.4. Limitations and Future Work

The following key limitations motivate further research:

- **Generality:** Single BTC–USD dataset; validation on additional assets (e.g., equities, commodities) and higher-frequency data is needed.
- **External benchmark scope:** The present study emphasizes a controlled within-family comparison (LSTM/SLSTM with and without GWO) rather than an exhaustive benchmark against stronger contemporary alternatives such as Transformer-based, decomposition-based, or hybrid attention models. Including such baselines is an important direction for future work.
- **Framework differences:** LSTM models were implemented in TensorFlow/Keras; SLSTM models were implemented in PyTorch/snnTorch. A unified framework would allow fairer cross-architecture comparison.
- **Energy and hardware:** Energy consumption and inference latency on neuromorphic chips were not measured; future work should benchmark on platforms like Intel Loihi or SpiNNaker.
- **Surrogate gradient sensitivity:** The choice of surrogate function (atan) and reset mechanism ('none') may affect the results; a systematic ablation study is recommended.
- **Ethical risk:** Financial forecasting models carry operational risk; any deployment in automated trading must include out-of-sample monitoring, fallbacks, and governance.

It should also be noted that the contribution of this study is primarily empirical and methodological: a controlled comparison of spiking-inspired recurrent dynamics and GWO-based optimization under a unified forecasting protocol, rather than the introduction of a fundamentally new theoretical forecasting framework.

7. Conclusions

This work evaluated four related configurations (LSTM, SLSTM, GWO-LSTM, and GWO-SLSTM) to determine the individual effects of spiking recurrence and Grey Wolf Optimizer-based hyperparameter tuning in non-stationary Bitcoin forecasting. On daily BTC–USD data (17 September 2014 to 9 October 2025), the key findings are as follows:

- **GWO-LSTM** achieved a test RMSE of 2217.24, test MAPE of 2.46%, and PBIAS of 0.11%, representing reductions of 45.0% in RMSE and 44.1% in MAPE relative to the baseline LSTM (RMSE 4030.10, MAPE 4.40%).
- **GWO-SLSTM** achieved a test RMSE of 1840.97, test MAPE of 1.76%, and PBIAS of 0.36%, representing reductions of 47.4% in RMSE and 54.4% in MAPE relative to the baseline SLSTM (RMSE 3501.48, MAPE 3.86%). In the present implementation, its training time was 39.71 s, lower than the 137.28 s observed for GWO-LSTM.

Within the present implementation, the spiking variant trained faster and achieved a better overall predictive accuracy, with a lower RMSE and MAPE than GWO-LSTM, although its PBIAS was slightly higher. However, this runtime difference should be interpreted cautiously because the LSTM and SLSTM model families were implemented in different frameworks and stopped after different numbers of epochs. The more defensible conclusion is therefore setup-specific: under the software and hardware conditions used in this study, GWO-SLSTM combined stronger predictive performance with a moderate and practically acceptable training cost.

Future work should extend evaluation to multiple assets, include stronger external baselines such as transformer-based or other contemporary forecasting models, unify the implementation framework, quantify energy consumption on neuromorphic hardware, and perform sensitivity analyses on surrogate gradient parameters. More broadly, this controlled experiment suggests that evolutionary search can be usefully paired with alternative

recurrent dynamics in non-stationary sequential forecasting, while broader claims about scalability, cross-asset generalization, and deployment efficiency require further validation.

Author Contributions: Conceptualization, F.N.W., M.W., M.M. and M.G.; methodology, F.N.W., M.W. and M.M.; software, F.N.W.; validation, F.N.W.; formal analysis, F.N.W. and M.G.; investigation, F.N.W.; data curation, F.N.W.; writing—original draft preparation, F.N.W.; writing—review and editing, M.W., M.M. and M.G.; supervision, M.G. and M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by ongoing national and EU-funded projects at the E9 Department of Jožef Stefan Institute and Alma Mater Europaea University. The authors also acknowledge funding from the Slovenian Research and Innovation Agency (ARIS), grant PR-10495, and basic core funding P2-0209.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw BTC–USD historical data used in this study are publicly available from Yahoo Finance via https://finance.yahoo.com/quote/BTC-USD/history/?p=BTC-USD (accessed on 18 January 2026). The access date format is ISO 8601 (also known as "YYYY-MM-DD" or "dash-separated year-month-day"). The code, processed datasets, train/test split indices, rolling-origin windows, and plotting scripts supporting the findings of this study are available at https://zenodo.org/records/18983179 (accessed on 18 March 2026).

Acknowledgments: The authors acknowledge Jožef Stefan Institute and the Applied Artificial Intelligence team at Alma Mater Europaea University for their mentorship and ongoing support throughout this research. During the preparation of this manuscript, the authors used ChatGBT large language model (LLM) for grammar, clarity, and linguistic refinement of the manuscript. A professional language editor recommended by the journal additionally reviewed and improved the text. The authors have reviewed and edited the output and take full responsibility for the content of this publication

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Descriptive Statistics, Rolling-Origin Evaluation, and Regime Distribution

Appendix A.1. Descriptive Statistics and Diagnostics

Table A1 shows the descriptive statistics and diagnostic test results (Augmented Dickey–Fuller (ADF) and Ljung–Box) for the training and testing subsets of daily Bitcoin closing prices (BTC–USD) spanning from September 2014 to October 2025. The results indicate strong non-stationarity and significant serial dependence, which justifies the use of adaptive temporal models.

Table A1. Descriptive statistics, ADF and Ljung–Box tests for daily BTC–USD closing prices (September 2014–October 2025).

Statistic	Train (80%)	Test (20%)
Mean	13,692.55	72,108.64
Std	16,015.91	28,526.86
Min	178.10	25,162.65
Max	67,566.83	124,752.53
Skew	1.38	0.02
Kurtosis	0.94	−1.17

Table A1. *Cont.*

Statistic	Train (80%)	Test (20%)
ADF statistic	−1.54	−0.53
ADF *p*-value	0.51	0.89
Ljung–Box stat	31,867.38	7711.74
Ljung–Box *p*-value	0.00	0.00

Appendix A.2. Market Regime Distribution

Figure A1 displays the distribution of market regimes (bull, bear, and neutral) based on daily returns, which categorize market states as bull ($R_t > 1\%$), bear ($R_t < -1\%$), or neutral. This distribution demonstrates balanced coverage of market conditions with 1345 bull days (33.3%), 1577 neutral days (39.0%), and 1118 bear days (27.7%).

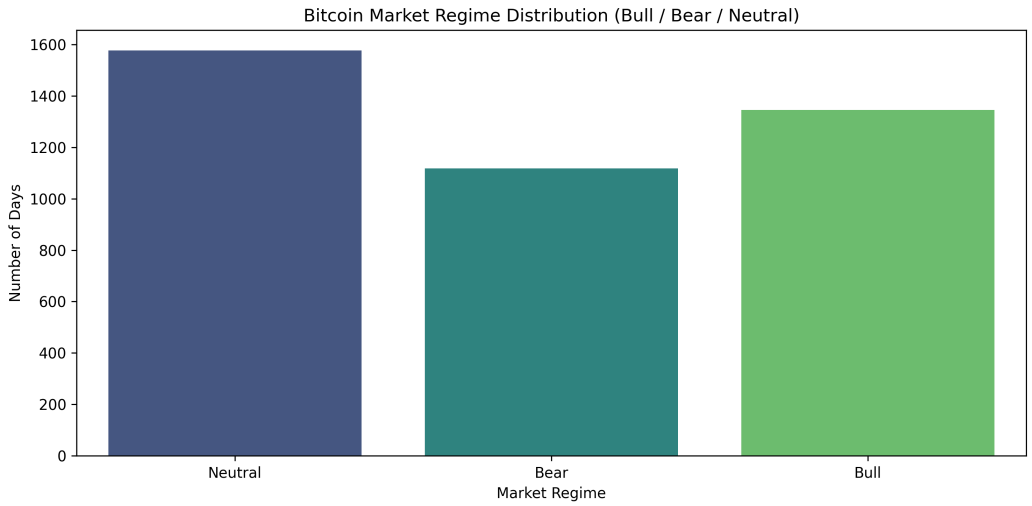

Figure A1. Bitcoin market regime distribution (bull, bear, neutral) based on daily returns (4040-day period, September 2014–October 2025). The distribution shows balanced representation across market conditions with 1345 bull days (33.3%), 1577 neutral days (39.0%), and 1118 bear days (27.7%).

Appendix B. Seven-Day Rolling

The rolling 7-day diagnostics in Figures A2–A4 and 8 further highlight robustness under regime shifts. Standard LSTM (Figure A2) shows a high average rolling RMSE (3365.55) and MAPE (4.40%), with a maximum RMSE of 11882.01. GWO-LSTM (Figure A3) significantly reduces these metrics (average RMSE 2003.80, average MAPE 2.45%; max RMSE 5241.01).

The SLSTM (Figure A4) exhibits moderate improvements over standard LSTM but remains less stable than GWO-LSTM (average RMSE 2975.79, average MAPE 3.86%; max RMSE 9781.52). In contrast, the GWO-SLSTM (Figure 8) exhibited the lowest rolling errors among the evaluated configurations, with the lowest average rolling RMSE (1624.47) and MAPE (1.76%), and a maximum RMSE of 4715.67—the smallest among all models, indicating lower rolling errors under the evaluated non-stationary conditions.

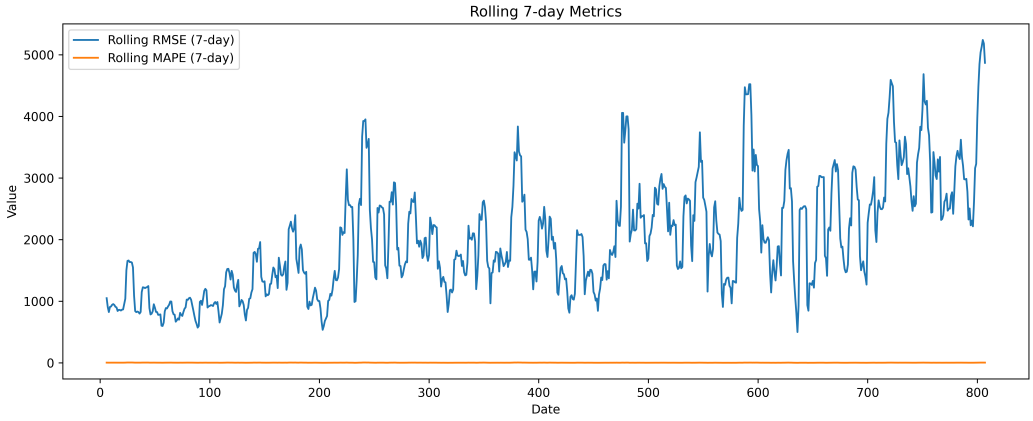

Figure A2. Seven-day rolling RMSE and MAPE for standard LSTM. Errors are substantial and variable (average RMSE = 3365.55, max = 11882.01; average MAPE = 4.40%, max = 15.00%), indicating high sensitivity to volatility.

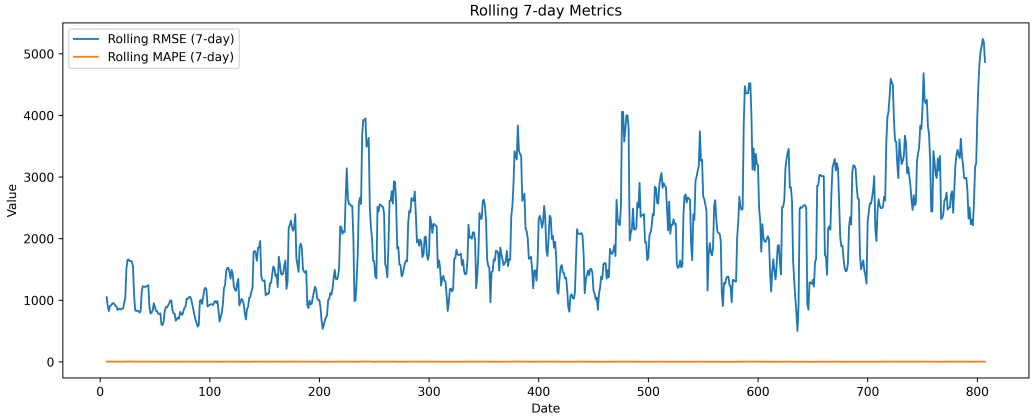

Figure A3. Seven-day rolling RMSE and MAPE for GWO-LSTM. The model shows significant improvement over standard LSTM (average RMSE = 2003.80, max = 5241.01; average MAPE = 2.45%, max = 5.77%).

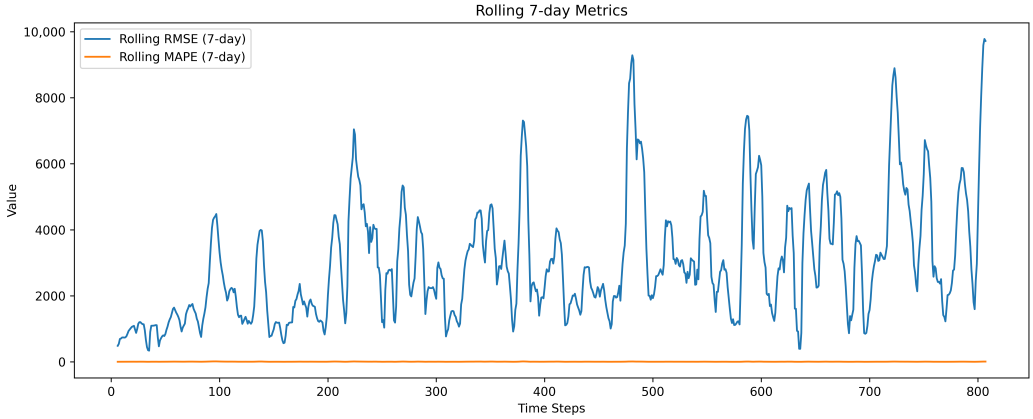

Figure A4. Seven-day rolling RMSE and MAPE for standard SLSTM. The spiking architecture provides moderate improvements (average RMSE = 2975.79, max = 9781.52; average MAPE = 3.86%, max = 12.86%).

Appendix C. Pseudocode of GWO-SLSTM

Nomenclature

Indices and Data:

t	Discrete time index.
x_t	Normalized input price at time t.
y_t	True Bitcoin closing price at time t (USD).
$\hat{y}_t$	Model-predicted Bitcoin closing price at time t (USD).
R_t	Daily return (percentage change) at time t.
$\mathcal{D}_{\text{train}}$	Training subset of the BTC–USD time series.
$\mathcal{D}_{\text{test}}$	Testing (hold-out) subset of the BTC–USD time series.
R1, R2, R3	Rolling-origin evaluation windows.

LSTM/SLSTM States:

c_t	LSTM cell state at time t.
h_t	LSTM hidden state at time t.
syn_t	SLSTM synaptic current at time t.
mem_t	SLSTM membrane potential at time t.
S_t	Binary spike output (0/1) at time t.

Spiking Parameters:

V_{th}	Membrane firing threshold.
V_{reset}	Reset membrane potential after a spike.
λ	Membrane decay factor for leaky integrate-and-fire dynamics.
σ_t	Noise gain in stochastic membrane dynamics.

Optimization and Hyperparameters:

η	Learning rate.
WD	Weight decay coefficient.
$\alpha, \beta, \delta, \omega$	Grey Wolf Optimizer leadership hierarchy levels.
$T_{\max}$	Maximum number of GWO iterations.
N	Number of observations in an evaluation set.

Evaluation Metrics:

MSE	Mean Squared Error.
RMSE	Root mean squared error.
MAPE	Mean absolute percentage error.
PBIAS	Percent bias.

Detailed explanations of the algorithms for SLSTM Cell Forward Pass with LIF Dynamics and Grey Wolf Optimizer for SLSTM Hyperparameter Search and Fitness Evaluation for GWO-SLSTM are provided as additional material via the following link only and are not part of the executed experimental pipeline reported in the main manuscript: https://zenodo.org/records/18983179 (accessed on 18 March 2026).

Appendix D. Future Methodological Extensions and Proposed Ablation Protocol

The following future methodological extensions and proposed ablation protocol are provided as supplementary material via the following link only and are not part of the executed experimental pipeline reported in the main manuscript: https://zenodo.org/records/18983179 (accessed on 18 March 2026).

Appendix E. Visualization Details

Additional diagnostic visualizations (radar charts, Taylor diagrams, and APE boxplots) were generated to cross-check agreement, variance matching, and error dispersion. These plots and their descriptions are provided as supplementary materials and can be downloaded via the following link: https://zenodo.org/records/18983179 (accessed on 18 March 2026).

References

1. Dinshaw, C.; Jain, R.; Hussain, S.A. Statistical Scrutiny of the Prediction Capability of Different Time Series Machine Learning Models in Forecasting Bitcoin Prices. In *Proceedings of the 2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA), Goa, India, 8–9 October 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 329–336. [CrossRef]
2. Yu, D. Cryptocurrency Price Prediction Based on Long-Term and Short-Term Integrated Learning. In *Proceedings of the 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 21–23 January 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 543–548. [CrossRef]
3. Abdulkadir, S.J.; Alhussian, H.; Nazmi, M.; Alwafi, A. Long short-term memory recurrent network for Standard and Poor's 500 Index modelling. *Int. J. Eng. Technol.* **2018**, *7*, 123–130. [CrossRef]
4. Gao,Y.; Wang, R.; Zhou, E. Stock Prediction Based on Optimized LSTM and GRU Models. *Sci. Program.* **2021**, *2021*, 4055281. [CrossRef]
5. Ito, K.; Iima, H.; Kitamura, Y. LSTM forecasting of foreign exchange rates using limit order book data. *Financ. Res. Lett.* **2022**, *47*, 102517. [CrossRef]
6. Vintha, N.; Devinder, K. Comparative Analysis of Deep Learning Approaches for Analysis and Prediction of Multivariate Time Series Data. In *Proceedings of the 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE), Las Vegas, NV, USA, 24–27 July 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 76–82. [CrossRef]
7. Skatchkovsky, N.; Jang, H.; Simeone, O. Spiking neural networks—Part II: Detecting spatio-temporal patterns. *IEEE Commun. Lett.* **2021**, *25*, 789–793. [CrossRef]
8. Ponghiran, W.; Roy, K. Spiking Neural Networks with Improved Inherent Recurrence Dynamics for Sequential Learning. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 8001–8009. [CrossRef]
9. Huang, Y.; Lin, X.; Ren, H.; Fu, H.; Zhou, Y.; Liu, Z.; Pan, B.; Cheng, B. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. *arXiv* **2024**, arXiv:2402.04663.
10. Zhang, Y.; Inoue, K.; Nakajima, M.; Hashimoto, T.; Kuniyoshi, Y.; Nakajima, K. Training spiking neural networks via augmented direct feedback alignment. *arXiv* **2024**, arXiv:2409.07776.
11. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
12. Fischer, T.; Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* **2018**, *270*, 654–669. [CrossRef]
13. Sen, A.; Dutta Choudhury, K. Forecasting the Crude Oil prices for last four decades using deep learning approach. *Resources Policy* **2024**, *88*, 104438. [CrossRef]
14. McNally, M.; Roche, J.; Caton, S. Predicting the price of Bitcoin using Machine Learning. In *Proceedings of the 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), Cambridge, UK, 21–23 March 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 339–343. [CrossRef]
15. Noh, Y. Deep Learning Model for Multivariate High-Frequency Time-Series Data: Financial Market Index Prediction. *Mathematics* **2023**, *11*, 3603. [CrossRef]
16. Zhang, W.; Lu, B. Stock Trend Prediction with Machine Learning: Incorporating Inter-Stock Correlation Information through Laplacian Matrix. *Big Data Cogn. Comput.* **2024**, *8*, 56. [CrossRef]
17. Tu, S.; Qin, P.; Zhu, M.; Zeng, Z.; Cheng, S.; Ye, B. Multi-Step Multidimensional Statistical Arbitrage Prediction Using PSO Deep-ConvLSTM: An Enhanced Approach for Forecasting Price Spreads. *Appl. Sci.* **2024**, *14*, 3798. [CrossRef]
18. Zhou, H.; Zhang, S.; Peng, J.; Huang, Y.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115. [CrossRef]
19. Zhang, Q.; Qin, C.; Zhang, Y.; Bao, F.; Zhang, C.; Liu, P. Transformer-based attention network for stock movement prediction. *Expert Syst. Appl.* **2022**, *202*, 117239. [CrossRef]
20. Qin, Y.; Zhong, Y.; Lei, Z.; Peng, H.; Zhou, F.; Tan, P. A Hybrid Parameter Estimation for Multi-Asset Modeling and Dynamic Allocation Based on Financial Market Microstructure Model. *Int. J. Artif. Intell. Tools* **2020**, *29*, 2040007. [CrossRef]

21. Qolomany, B.; Maabreh, M.; Al-Fuqaha, A.; Gupta, A.; Benhaddou, D. Parameters optimization of deep learning models using particle swarm optimization. In *Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1285–1290. [CrossRef]
22. Zeng, X.; Liang, C.; Yang, Q.; Wang, F.; Cai, J. Enhancing stock index prediction: A hybrid LSTM–PSO model for improved forecasting accuracy. *PLoS ONE* **2025**, *20*, e0310296. [CrossRef] [PubMed]
23. Mirjalili, S. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
24. Khayat, A.; Kissaoui, M.; Bahatti, L.; Raihani, A.; Errakkas, K.; Atifi, Y. Efficient day-ahead energy forecasting for microgrids using LSTM optimized by grey wolf algorithm. *e-Prime—Adv. Electr. Eng. Electron. Energy* **2025**, *13*, 101054. [CrossRef]
25. Liu, T. A comparative study of transformer-based and classical models for financial time-series forecasting. *J. Risk Financial Manag.* **2026**, *19*, 203. [CrossRef]
26. Behera, S.; Nayak, S.C.; Kumar, A.V.S.P. A Comprehensive Survey on Higher Order Neural Networks and Evolutionary Optimization Learning Algorithms in Financial Time Series Forecasting. *Arch. Comput. Methods Eng.* **2023**, *30*, 4401–4448. [CrossRef]
27. Tian, C.; Li, R. Predicting Bond Defaults in China: A Double-Ensemble Model Leveraging SMOTE for Class Imbalance. *Big Data Cogn. Comput.* **2026**, *10*, 81. [CrossRef]
28. Gülmez, B. Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm. *Expert Syst. Appl.* **2023**, *227*, 120346. [CrossRef]
29. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. [CrossRef]
30. Neftci, E.O.; Mostafa, H.; Zenke, F. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks *IEEE Signal Process. Mag.* **2019**, *36*, 61–76. [CrossRef]
31. Bellec, G.; Scherr, F.; Subramoney, A.; Hajek, E.; Salaj, D.; Legenstein, R.; Maass, W. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* **2020**, *11*, 3625. [CrossRef] [PubMed]
32. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Netw.* **2019**, *111*, 47–63. [CrossRef] [PubMed]
33. Sankaran, A.; Detterer, P.; Kannan, K.; Alachiotis, N.; Corradi, F. An Event-Driven Recurrent Spiking Neural Network Architecture for Efficient Inference on FPGA. In *Proceedings of the ICONS '22: Proceedings of the International Conference on Neuromorphic Systems 2022*; ACM: New York, NY, USA, 2022; p. 12. [CrossRef]
34. Molaei, S.; Cirillo, S.; Solimando, G. Cancer Detection Using a New Hybrid Method Based on Pattern Recognition in MicroRNAs Combining Particle Swarm Optimization Algorithm and Artificial Neural Network. *Big Data Cogn. Comput.* **2024**, *8*, 33. [CrossRef]
35. Torre-Bastida, A.I.; Díaz-de-Arcaya, J.; Osaba, E.; Del Ser, I.; Bilbao, M.N.; Santamaría, A.D. Bio-inspired computation for big data fusion, storage, processing, learning and visualization: State of the art and future directions. *Neural Comput. Appl.* **2021**, *33*, 6119–6138. [CrossRef]
36. Gams, M.; Horvat, T.; Kolar, Ž.; Kocuvan, P.; Mishev, K.; Misheva, M.S. Evaluating a Nationally Localized AI Chatbot for Personalized Primary Care Guidance: Insights from the HomeDOCtor Deployment in Slovenia. *Healthcare* **2025**, *13*, 1843. [CrossRef] [PubMed]
37. Chen, M.; Fortino, G. Big Data and Cognitive Computing: Five New Journal Sections Established. *Big Data Cogn. Comput.* **2026**, *10*, 26. [CrossRef]
38. Zhang, Y.; Gao, Y.; Huang, L.; Xie, X. An Improved Grey Wolf Optimizer Based on Attention Mechanism for Solving Engineering Design Problems. *Symmetry* **2025**, *17*, 50. [CrossRef]
39. Faris, H.; Mirjalili, S.; Aljarah, I. Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 2901–2920. [CrossRef]
40. Yang, Z. Competing leaders grey wolf optimizer and its application for training multi-layer perceptron classifier. *Expert Syst. Appl.* **2024**, *239*, 122349. [CrossRef]
41. Samantaray, S. Prediction of Rainfall Using Hybrid BPNN–PSO in Jhelum River Basin: A Case Study. In *Proceedings of International Conference on Information Technology and Intelligence: ICITI 2024, Volume 1, Lecture Notes in Networks and Systems*; Sharma, H., Chakravorty, A., Eds.; Springer: Singapore, 2025; Volume 1342, pp. 139–151. [CrossRef]
42. Samantaray, S.; Sahoo, A. Groundwater level prediction using an improved ELM model integrated with hybrid particle swarm optimisation and grey wolf optimisation. *Groundw. Sustain. Dev.* **2024**, *26*, 101178. [CrossRef]
43. Samantaray, S.; Sahoo, A. Prediction of flow discharge in Mahanadi River Basin, India, based on novel hybrid SVM approaches. *Environ. Dev. Sustain.* **2024**, *26*, 14969–14991. [CrossRef]
44. Mahakur, V.; Mahakur, V.K.; Samantaray, S.; Ghose, D.K. Prediction of runoff at ungauged areas employing interpolation techniques and deep learning algorithm. *HydroResearch* **2025**, *8*, 265–275. [CrossRef]

45. Samal, P.; Patnaik, P.; Samantaray, S.; Swain, P.C. Comparative study of reservoir operations using TLBO, PSO and DE optimization techniques: An experiment on the Hirakud reservoir, Odisha, India. *J. Water Clim. Change* **2024**, *15*, 3133–3152. [CrossRef]
46. Mabrouk, A.; Inam, I.; Qureshi, M.Z.; Ali, T.; Murtaza, N.; Ouda, M.M.; Alawi Al-Naghi, A.A.; Tasán Cruz, D. Artificial intelligence evaluation of nature-based flood resilience in hilly terrain. *Sci. Rep.* **2025**, *15*, 35492. [CrossRef] [PubMed]