# Lifting the Veil of Non-Stationarity in Financial Market

**Tianhao Fu,*, Xinxin Xu,*Xuanmeng Zhang, Weichen Xu, Ruilong Ren, Bowen Deng, Xinyu Zhao, Jian Cao, Xixin Cao**

Peking University

## Abstract

Financial asset price movement prediction is inherently challenging due to the non-stationary nature of financial markets, where data distributions shift over time. Existing methods often assume that the market is stationary, which limits their applicability. To address this, we propose the Market-State Jump Diffusion Framework(MSJD), which models non-stationarity through two key components: an Explicit Market-State Jump Diffusion Process(EMJD) and an Implicit Market-State Jump Diffusion Process(IMJD). EMJD captures the dynamics of diffusion, drift, and jump processes governed by latent market states, formulated as stochastic differential equations, to explicitly model non-stationarity and solved via neural networks. IMJD integrates these components into a multi-modal large language model, enabling interpretable predictions across varying market conditions through temporal point encoding and jump diffusion embeddings to learn the non-stationary implicitly. Additionally, we introduce a general modality synthesizer that employs a unified adversarial masking strategy to complete missing modalities and fine-tune the prediction model. Extensive experiments on real-world stock and cryptocurrency datasets demonstrate that our method significantly outperforms existing approaches in the prediction of price movements. The code is available in the supplementary material.

## Introduction

Forecasting the price movement of a financial asset over a specific period is a crucial task in economic analysis.(Lewellen 2004). The global research community has recently recognized the potential of deep neural networks in the prediction of price movement. (Martinez-Miera and Repullo 2019; Sezer, Gudelek, and Ozbayoglu 2020). However, these methods are typically trained to make predictions in an assumed stationary environment (Ogasawara et al. 2010; Passalis et al. 2019; Kim et al. 2021), where the data distribution is fixed over time.

In practice, stationary environments are rare. As shown in Figure 1, real financial market environments are dynamic and non-stationary (Li et al. 2014), which means the statistical process and joint distribution often change over time with different change types and make the time series less predictable. For example, momentum characteristics and price changes show a positive correlation in trending markets but a negative correlation in reversal markets (Avellaneda and Stoikov 2008). This suggests that the role of eigenvalues is inextricably linked to the market environment in which data are embedded, while the financial environment changes rapidly alongside breaking news and global events (Guéant, Lehalle, and Fernandez-Tapia 2013; Graves and Graves 2012; Vaswani 2017).

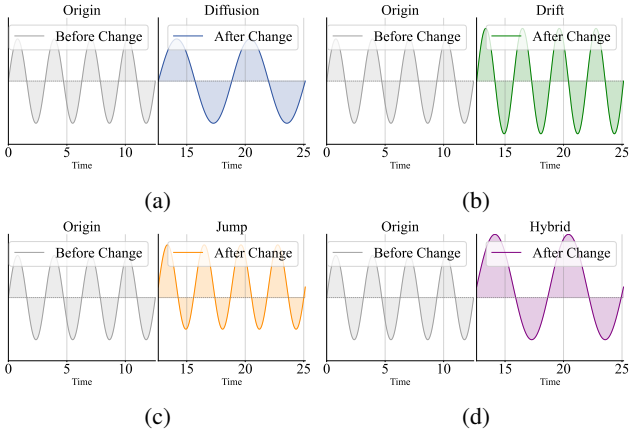


Figure 1: Different types of change in price series statistical properties and joint distribution over time. Through the Diffusion/Drift/Jump process, we could formulate various distribution transformations.

Some work has attempted to use neural networks to detect environmental changes in time series (Anderson 1976) to learn price characteristics in nonstationary environments (Shazeer et al. 2017; Hyndman and Athanasopoulos 2018; Brown et al. 2020; Sawhney et al. 2020; Yang et al. 2020). However, environmental changes in the market often stem from a variety of causes, some of which may be reflected in time series data, others in news text or images of a company spokesperson's press conference (Brown et al. 2020). Some recent attempts have been made to make price forecasts based on multimodal data (Sawhney et al. 2020; Yang et al. 2020). However, these efforts have overlooked two critical issues: (1) Issue of the single-modal method: Environmental changes in the market often stem from a variety of causes,

*These authors contributed equally.

some of which may be reflected in time-series data, others in news text or images of a company spokesperson's press conference. The single-modal (time series-based method) omits some vital information. (2) Issue of multi-modal methods: The current multi-modal method is an end-to-end method, which lacks interpretability(Hinton 2015) and often finds a problem of missing modalities(Cho and Hariharan 2019). The former problem means we do not know exactly which characteristics of the stock correspond to each layer of these models and how much weight they carry in the final price prediction. Once the model performance came across performance decay in real trading, we cannot do traceability analysis. The latter problem arises because, in some real-time scenarios, modalities may be missing due to issues like the loss of text data or an outage of the server storing a particular modality.

Inspired by recent progress in multi-modal LLMs and stochastic process models of finance (Zhang et al. 2024a; Jacobs et al. 1991), we propose the Market-State Jump Diffusion Framework (MSJD). In detail, MSJD consists of four components: (1) We present a new stochastic process to model the price changes of financial assets that is controlled by a hidden variable of the market state and can directly model different components of the price environments of financial assets, including trend diffusion processes, trend drift processes, and jump processes (Costabile et al. 2014; Bertsekas 1996). Each process contributes to the stable or unstable components of the environment. (2) We make explicit and implicit realizations of such stochastic processes using neural differential equations and multi-modal LLMs, and integrate both into a single framework. (3) The three differential terms of the process of changing from the explicit solution of the neural differential equations are integrated into the positional encoding, embedding, and attention of the multimodal LLM, as well as the final output of the logits, thus obtaining interpretable price forecasts in different market environments. (4) A general modality synthesizer uses a unified adversarial masking post-training strategy(Mizrahi et al. 2023) to migrate a multimodal LLM from the general domain to the financial domain and further improve its performance. Having seen the patterns of change in financial assets in a wide variety of market environments, the synthesizer is able to model the full range of financial market world models based on a limited number of modalities and show the financial asset market in a certain state in various modes, thus completing the effect of missing modality (Lin et al. 2024).

Finally, we conduct extensive experiments on some publicly datasets, in which our method outperforms the best baseline across all metrics for both price movement and volatility prediction. In summary, the contributions of this paper are as follows:

- We propose a market state dominated statistic process and a corresponding neural framework with a neural network SDE to explicitly model the process and implicitly encapsulate the process into a Multi-Modal LLM. Such a framework could deal with a non-stationary environment.
- We propose a general modality synthesizer with a unified adversarial masking post-training method. The synthesizer could learn different modality interaction rules from the financial world to complete the missing modality, which is useful for our statistical process.
- We conducted extensive experiments on six real-world datasets, including stocks and crypto, which verify the effectiveness of our method on price movement prediction tasks, together with high accuracy and trading profitability.

## Related Works

**Non-Stationary Environment.** Time series forecasting often assumes stationarity (Paparoditis 2010), but real-world data frequently violates this. Classical models like ARIMA (Shumway et al. 2017) use differencing, while deep learning methods apply normalization techniques such as DAIN (Passalis et al. 2019) and RevIN (Kim et al. 2021). Compared with these methods, we detect non-stationarity not only in time series, but also from other modalities.

**Neural Temporal Point Processes.** Neural TPPs (Daley and Vere-Jones 2006; Yang, Mei, and Eisner 2021) enhance classical TPPs but often assume fixed intensity forms, e.g., exponential (RMTPP (Du et al. 2016)) or softplus (THP (Zuo et al. 2020)). Such assumptions limit flexibility. Some alternatives model cumulative (Omi, Aihara et al. 2019) or conditional density functions (Shchur, Biloš, and Günnemann 2019), but still fall short. Compared with these methods, we model the process with a latent financial market variable which could reflect each process's market state.

**Price Movement Forecasting.** Forecasting methods either enrich input data (Ozbayoglu, Gudelek, and Sezer 2020; Gu, Kelly, and Xiu 2020) or develop specialized models (Albuquerque, Peng, and Silva 2022; Sui et al. 2024). Compared with these methods, our prediction considers non-stationarity.

## Method

In this section, we first propose a new explicit market jump diffusion process to model the environment, which contains the environment's stationary component and non-stationary component that depends on the market state. We use a neural stochastic differential equation to implement the process and compute each component directly. Next, we show how we use these components to enhance our implicit market jump diffusion process from a different perspective, enabling our method could address a nonstationary environment. Figure 2 is the whole framework overview.

**Explicit Market-State Jump Diffusion Process.** The Market-State Jump Diffusion Process models the market state as a superposition of three fundamental statistical processes: diffusion, drift, and jump. Mathematically, the dynamics of the price $S(t)$ can be described by the following stochastic differential equation (SDE):

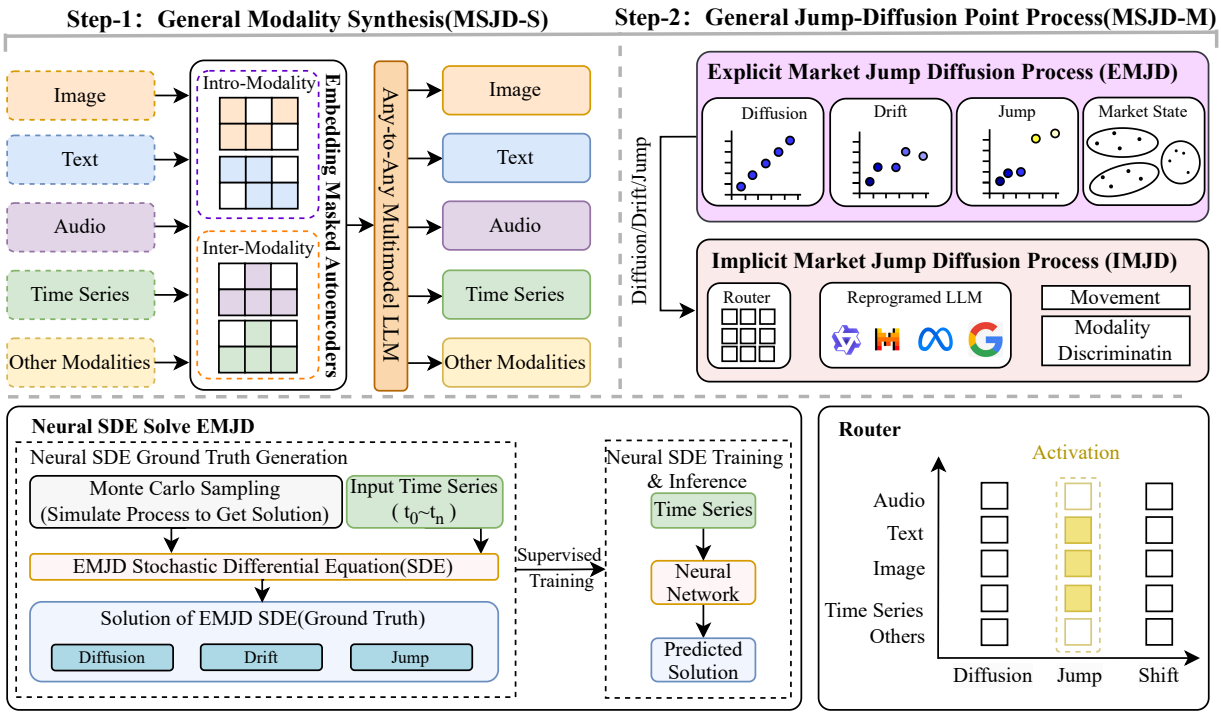


Figure 2: Overview of the Market-State Jump Diffusion (MSJD) Framework. The MSJD-S module first completes any missing data modalities. These complete data are then input into the MSJD-M module, which consists of two components: an Explicit Market Jump Diffusion (EMJD) model and an Implicit Market Jump Diffusion (IMJD) model. The EMJD utilizes a Neural SDE to compute the drift, diffusion, and jump terms of the time series. The IMJD subsequently uses these terms to reprogram a Large Language Model for the final prediction.

$$\begin{aligned} dS(t) &= \mu S(t, M(t))dt \\ &+ \sigma S(t, M(t))dW(t) \\ &+ J(t, M(t))dN(t) \end{aligned} \tag{1}$$

Where $\mu$ represents the drift coefficient, $\sigma$ denotes the diffusion coefficient, $M(t)$ is a Market state latent variable, $W(t)$ is a standard diffusion process, $J(t)$ is the jump size, and $N(t)$ is a Poisson process with intensity $\lambda$. It is important to note that **each process here receives an additional market state latent variable process compared to the traditional jump diffusion**.

To solve the corresponding Stochastic Differential Equation (SDE) associated with the stochastic process, we employ the Monte Carlo simulation method. The SDE governing the evolution of the probability density function $p(S,t)$ is given by:

$$\begin{aligned} &\frac{\partial p}{\partial t} + \mu S \frac{\partial p}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 p}{\partial S^2} \\ &= \lambda \int_0^\infty [p(S/J,t) - p(S,t)] f(J) dJ \end{aligned} \tag{2}$$

where $f(J)$ is the probability density function of the jump size $J$.

By discretizing this SDE and applying the Monte Carlo simulations, we generate labels corresponding to each of the three statistical processes for the time series data. These labels serve as supervisory signals for training the neural network, enabling it to learn the underlying market dynamics effectively. The reason why we do not use Monte Carlo simulations in inference is their low speed. So we utilize the trained neural network to obtain the solution of the SDE in the following steps.

**Implicit Market-State Jump Diffusion Process.** The explicit market jump diffusion process solution is integrated into the implicit market state jump diffusion process(IMJD), which contains several components: router, temporal point position encoding, and jump diffusion embedding mechanisms. The central idea of IMJD is to incorporate jump diffusion information when post-training multimodal LLMs in an end-to-end training process as shown in Figure 3.

**Router.** In fitting a stochastic process for stocks based on various modes, for a given data sample, some models help predict price movement, while some models are noise that is not helpful for price prediction. In this case, we need to choose from different modals. Similarly, when selecting a stochastic process for modeling particular data, the com-

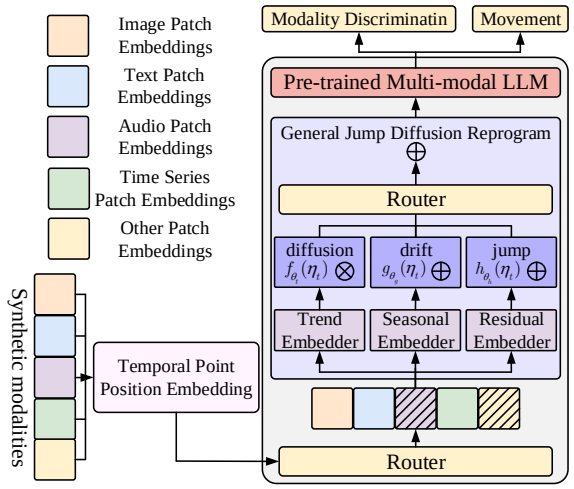


Figure 3: Implicit Market-State Jump Diffusion Process. This model uses a multi-modal LLM to represent the market-state jump diffusion process implicitly. Firstly, temporal point position embedding aligns all modalities along the time dimension. Then the router selects effective modalities to compute a mean embedding. Such embedding is decomposed into trend, seasonal, and residual components via time series analysis, where the token sequence serves as the time dimension. These components is modulated by the drift, diffusion, and jump terms respectively. The router recomposes these elements into a reprogrammed embedding and then does multi-modal LLM normal computation.

plexity of the stochastic process varies depending on the data. For example, most stocks do not have a jump process most of the time, and stocks with extremely low prices close to delisting do not have a drift process most of the time. In such a case, how to choose the appropriate stochastic process to model the asset is also an important issue. In many cases, the choice of modality and the choice of process are interrelated events. For instance, time series typically correspond to diffusion processes, whereas image/text/audio data generally correspond to drift/jump processes. In these two scenarios, we have designed a two-dimensional router to select the appropriate modal and stochastic processes to predict price fluctuations based on different data types.

The two-dimensional router is mathematically represented as:

$$S(t) = \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_{i,j}(t) \mathcal{R}(\mathcal{M}_i, \mathcal{P}_j) \tag{3}$$

where $\sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_{i,j}(t) = 1$ and $\alpha_{i,j}(t) \geq 0$ for all $i$ and $j$. Here, $\alpha_{i,j}(t)$ represents the routing weight for the $i$-th modality $\mathcal{M}_i$ and the $j$-th stochastic process $\mathcal{P}_j$. In practice, we implement the router with a shallow neural network that inputs different modalities or random process embedding and outputs a selective vector that is supervised in an end-to-end manner.

**Temporal Point Position Encoding.** To effectively capture time-related features, we extend traditional positional encoding by incorporating temporal point position encoding. The final position encoding is computed by adding each token real datetime to the token's origin positional encoding. In this way, we can align the embedding of different modalities in the time dimension.

**Jump Diffusion Embedding.** We first compute the overall embedding by taking the mean embeddings from various router-selected modalities, including images, text, audio, and time series. Each modality embedding comes from embedding layers.

Subsequently, we use the traditional plus-based time series decomposition method to decompose the time series embedding into its constituent components: trend, seasonal, and residual embedding. We could easily observe that the trend, seasonal, and residual natural correspond exactly to diffusion, drift, and jump processes. Therefore, we have multiple trend embeddings with diffusion, seasonal embeddings with drift, and residual embeddings with jump. In this way, the jump diffusion information has been ingeniously incorporated into the multi-modal LLM.

## General Modality Synthesizer

In this section, we introduce the **General Modality Synthesizer (GMS/MSJD-S)**, designed to integrate and synthesize data from various modalities such as images, text, audio, and time series. The GMS that can effectively complete missing modalities can be initialized with any multi-modal LLM outside the finance domain. To further improve the **GMS** performance in the price movement task, we propose a unified adversarial masking post-training method consisting of two steps: (1) self-supervised mask-based training, and (2) fine-tuning using the synthesizer as a discriminator and an implicit-explicit jump diffusion process model as a generator in an adversarial training way.

**Mask Post-Training.** This approach encompasses multiple masking operations from various perspectives, including the input/target token view and the inner/inter-modality token view. We first randomly select the input and target tokens from each modality, ensuring no overlap between them, as illustrated by the inner-modality mask strategy. Next, to enhance the consistency between GMS-generated modalities, we perform an inter-modality mask strategy. The specific steps are shown in Algorithm 1. In this step, for the input token that has been selected for a certain modality, we select the input token that is similar to it in the embedding space of the other modalities, as indicated in lines 3 and 10 to 12. Next, we perform the same operation for the target tokens as described in lines 4 and 14 to 16. The similarity between tokens across modalities is calculated using the Pearson correlation coefficient. These semantically similar tokens across modalities generally characterize the same information, e.g., a rising price token in the text often corresponds to a rising trend token in the image and a

happier mood token in the speech. In this post-training, we enable GMS to learn the relationships between the modalities, which can be used to complete the modality.

Algorithm 1: Inner/Inter-Modality Mask Strategy

**Input:** Input tokens for each modality $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$, where $I_i$ represents different modality input token generated by encoder; reconstruction target tokens for each modality $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, where $T_i$ represents different modality target generated by decoder. masking ratio $\mu$; correlation threshold $\tau$;

**Output:** Selected Input tokens $\mathcal{X}_{input}$ and Selected Target tokens $\mathcal{X}_{target}$ for each modality.

**Inner-Modality Mask Strategy:**
**for** each modality $j \in \mathcal{N}$ **do**
  Randomly select a subset $\mathcal{X}_j^{input} \subseteq I_j$ such that $|\mathcal{X}_j^{input}| = \mu_{intra} \cdot |I_j|$.
  Randomly select a subset $\mathcal{X}_j^{target} \subseteq T_j$ such that $|\mathcal{X}_j^{target}| = \mu_{intra} \cdot |T_j|$ and $\mathcal{X}_j^{input} \cap \mathcal{X}_j^{target} = \emptyset$.
**end for**
**Inter-Modality Mask Strategy:**
Let $\mathcal{Y}_i^{input} = I_i \setminus \mathcal{X}_i^{input}$ be the unselected input tokens in $I_i^{input}$ and Let $\mathcal{Y}_i^{output} = I_i^{target} \setminus \mathcal{X}_i^{output}$ be the unselected target tokens in $T_i^{target}$.
**for** each modality $j \in \mathcal{N}$ **do**
  **for** each token $y_j \in \mathcal{Y}_j$ **do**
    Compute correlation scores $corr(x_j^{input}, y_i^{input})$ for all $x_i^{input} \in \mathcal{X}_i^{input}$.
    **if** $\min x_i^{input} \in \mathcal{X}_i^{input}, s(x_j^{input}, y_i^{input}) > \tau$ **then**
      ADD $y_j^{input}$ to $\mathcal{X}_j^{input}$.
    **end if**
    Compute correlation scores $corr(x_j^{target}, y_i^{target})$ for all $y_i^{target} \in \mathcal{Y}_i^{target}$.
    **if** $\min x_i^{target} \in \mathcal{X}_i^{target}, s(x_j^{target}, y_i^{target}) > \tau$ **then**
      Add $y_j^{target}$ from $\mathcal{X}_j^{target}$.
    **end if**
  **end for**
**end for**
**return** Selected Input tokens $\mathcal{X}_{input}$ and Selected Target tokens $\mathcal{X}_{target}$ for each modality.

**Adversarial Post-Training.** After mask post-training, our GMS already has a good ability to generate multimodality in the financial domain. In order to further strengthen the effect, we let the IMJD additionally output a vector. Each dimension of the vector represents whether each modality input to the IMJD is real data or generated, respectively. In this way, we treat the IMJD as a discriminator and the GMS as a generator, and the two independent multimodal models fight against each other to further fine-tune the GMS so that it can evolve to achieve better results.

## Training and Optimization

The training process is divided into two steps. In the first step, the MSJD-S(synthesizer) is trained through mask post-training. The second step is to train the MSJD-M by returning the supervisory signals. Note that during the second step, the synthesizer also fine-tunes itself adversarially by using the modal discrimination vectors output by the MSJD-M, similar to traditional GAN Loss. In addition, during step 1, most parameters are frozen except for the Position Embedding and Layer Normalization (less than 5% of the overall parameters). The overall loss function $\mathcal{L}$ of MSJD-M in step-22 comprises two components: Target reconstruction loss $\mathcal{L}_{\text{recon}}$ and the modality gan loss. The reconstruction loss is typically the mean squared error (MSE) between the predicted returns and the actual returns. Notably, we could also make MSJD-M output an uncertainty score. Details refer to supplementary material.

# Experiments

In this section, we present extensive experiments to answer the following questions. Q1: How does MSJD perform in predicting movements? Q2: How do key components contribute to the performance of MSJD? Q3: Can MSJD adapt to non-stationary environments? Q4: How does EMJD outperform traditional stochastic processes? Q5: How effective is the GMS in handling missing modalities?

## Experimental Setup

**Multi-Modal Dataset.** Follow FinAgent (Zhang et al. 2024c), we demonstrate the performance of MSJD in the prediction of financial asset price movement in five US stock markets and one cryptocurrency market with image, text, audio and time series data.

**Evaluation metrics.** We compare MSJD with other state-of-the-art (SOTA) methods in terms of the following 6 financial metrics, which include 1 profit metric: annual return rate (ARR) (Fama and French 1993), 3 risk adjusted profit metrics: SR (Sharpe 1966), Calmar ratio (CR) (Vyachkileva 2018), Sortino ratio (SOR) (Sortino and Price 1994), and 2 risk metrics: maximum drawdown (MDD) (Chekhlov, Uryasev, and Zabarankin 2005), volatility (VOL) (Black and Scholes 1973). It is important to note that our method outputs a price movement and not a trading strategy, so for comparison with other multimodal LLM investment strategy methods, we form an investment strategy based on our movement prediction, used in conjunction with the simplest daily frequency long-short method.

**Comparative Methods.** To evaluate our multi-agent trading framework, we compare our performance with traditional trading strategies and advanced algorithmic approaches. Including (1) Rule-based: Buy-and-Hold (B&H) (Fama and French 1992), Moving Average Convergence Divergence (MACD) (Vaidya 2020), LSTM (Hochreiter 1997), (2) DL&RL-based methods: LGBM (Ke et al. 2017), Transformer (Vaswani 2017), PPO (Schulman et al. 2017). and (3) LLM based methods: FinGPT (Yang, Liu, and Wang 2023),

| Categories | Models | AAPL | | | AMZN | | | GOOGL | | | MSFT | | | TSLA | | | ETHUSD | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ARR↑ | SR↑ | MDD↓ | ARR↑ | SR↑ | MDD↓ | ARR↑ | SR↑ | MDD↓ | ARR↑ | SR↑ | MDD↓ | ARR↑ | SR↑ | MDD↓ | ARR↑ | SR↑ | MDD↓ |
| Market | B&H | 13.0 | 0.6 | 14.78 | 42.33 | 1.08 | 17.38 | 22.47 | 0.71 | 12.97 | 22.49 | 0.84 | 12.97 | 37.4 | 0.72 | 32.65 | 29.26 | 0.87 | 23.21 |
| Rule-based | MACD | 11.86 | 0.72 | 10.38 | 14.27 | 0.71 | 7.84 | -18.0 | -0.89 | 20.07 | 15.23 | 0.77 | 8.34 | -4.9 | -0.02 | 14.15 | 10.24 | 0.47 | 24.32 |
| | KDJ&RSI | 2.17 | 0.17 | 11.88 | 19.38 | 0.65 | 17.27 | 24.39 | **2.13** | **2.03** | 18.84 | 1.06 | 7.78 | 2.14 | 0.17 | 24.73 | 8.87 | 0.51 | 16.95 |
| | ZMR | -3.91 | -0.22 | 8.88 | 18.73 | 0.84 | 7.89 | 32.51 | 1.45 | 5.38 | 9.86 | 0.71 | 6.22 | -7.28 | -0.09 | 19.9 | 29.35 | 1.23 | 13.11 |
| ML&DL-based | LGBM | 16.93 | 1.47 | **2.52** | 29.34 | 0.72 | 17.41 | 24.77 | 0.7 | 12.98 | 19.28 | 0.67 | 12.96 | 15.57 | 0.84 | **3.88** | 24.91 | 0.72 | 22.96 |
| | LSTM | 10.97 | 0.54 | 11.95 | 15.91 | 0.64 | 17.41 | 18.97 | 0.6 | 15.75 | 16.48 | 0.64 | 11.75 | 17.36 | 0.78 | 4.44 | 36.09 | 1.03 | 21.5 |
| | Transformer | 17.11 | 0.96 | 7.53 | 32.66 | 1.11 | **4.96** | 13.69 | 0.46 | 14.36 | 17.44 | 1.46 | **2.59** | 39.7 | 1.04 | 8.17 | 31.0 | 1.02 | 12.93 |
| RL-based | DQN | 7.92 | 0.04 | 14.88 | 27.43 | 1.17 | 5.27 | 34.4 | 1.39 | 11.84 | 19.86 | 1.02 | 5.67 | 31.0 | 0.84 | 28.12 | 29.81 | 1.18 | 9.53 |
| | PPO | 13.26 | 0.61 | 14.78 | 23.83 | 0.64 | 16.89 | 28.1 | 1.22 | 15.96 | 20.22 | 0.87 | 8.46 | 33.64 | 0.74 | 15.37 | 25.94 | 0.31 | 11.12 |
| LLM-based | FinGPT | -5.46 | -0.17 | 16.23 | 42.93 | 1.03 | 18.9 | 12.24 | 0.48 | 12.86 | 21.07 | 0.68 | 9.84 | 38.43 | 0.75 | 21.57 | 21.56 | 0.68 | 25.56 |
| | FinMem | 23.78 | 1.11 | 10.39 | 31.6 | 0.97 | 10.0 | 25.61 | 1.45 | 8.14 | 25.1 | 1.09 | 7.46 | 50.04 | 0.92 | 9.62 | 44.72 | 1.27 | 15.59 |
| | FinAgent | 31.9 | 1.43 | 10.4 | 65.1 | 1.61 | 12.14 | 56.15 | 1.78 | 8.45 | 44.74 | 1.79 | 5.57 | 92.27 | 2.0 | 12.14 | 43.08 | 1.8 | 12.72 |
| | FinVision | 14.79 | 1.20 | 14.38 | 42.14 | 1.72 | 12.09 | - | - | - | 25.57 | 1.41 | 13.28 | - | - | - | - | - | - |
| | FinCon | 27.35 | 1.7 | 15.27 | 24.85 | 0.90 | 25.89 | 25.08 | 1.05 | 17.53 | 31.63 | 1.54 | 15.01 | 82.87 | 1.97 | 29.73 | - | - | - |
| Ours | **MSJD** | **35.6** | **1.86** | 8.6 | **68.8** | **1.91** | 10.38 | **59.63** | 1.97 | 8.15 | **46.66** | **1.93** | 4.00 | **95.14** | **2.25** | 11.34 | **46.01** | **2.14** | **9.05** |

Table 1: Performance comparison of all methods on six different datasets.

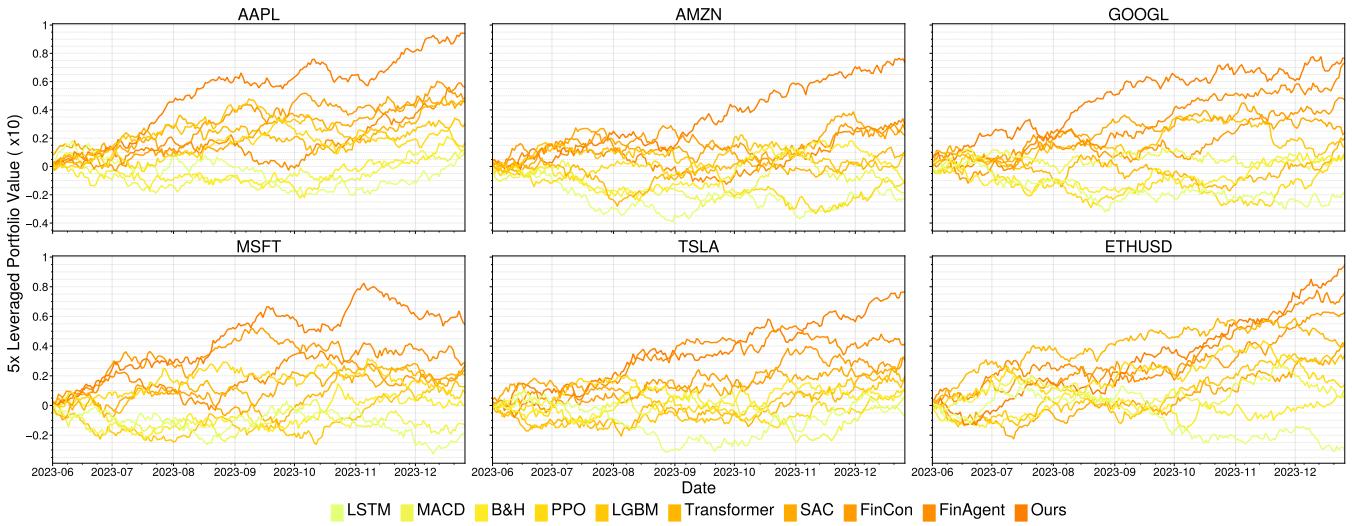


Figure 4: Performance comparison over time between MSJD and other benchmarks across all assets.

FinMem (Yu et al. 2024), FinAgent (Zhang et al. 2024c) and FinVision (Fatemi and Hu 2024).

**Implementation Detail.** We initialize our MSJD-S and MSJD-M using Next-GPT (Wu et al. 2023) coupled with Time-LLM to process time series, note that the two models MSJD-S and MSJD-M do not share weights. We use NJDSDE (Zhang et al. 2024b) to solve our proposed EMJD with an additional market state variable, which is trained with a traditional Monte Carlo solution label. For the MSJD-S post-training procedure, follow the 4M (Mizrahi et al. 2023), we set the mask ratio to 0.5, with 50 epoch. For the MSJD-M train procedure we set the epoch to 72. All experiments are conducted on 64 NVIDIA H100 GPUs.

**Comparison with Baselines (Q1).** As illustrated in Table 1 and Figure 4, the MSJD framework demonstrates superior performance in all evaluation metrics compared to other SOTA methods. Notably, MSJD achieves an impressive 95.14% ARR on the TSLA dataset, accompanied by a SR of 2.25. This performance outperforms FinAgent on the same dataset. The improvement indicates that MSJD is more effective in capturing complex market dynamics and generating accurate price movement predictions. Furthermore, MSJD exhibits robust risk management capabilities, with MDD consistently below 12% across various stocks. This represents a reduction of approximately 20% compared to transformer-based approaches, highlighting the framework's ability to mitigate risks effectively in volatile market conditions. For the ETHUSD dataset, MSJD achieves an ARR of 46.01% and an MDD of 9.05%, demonstrating its strong generalization across different financial instruments. MSJD's consistent outperformance across multiple datasets demonstrates its robustness and adaptability.

**Effectiveness of Each Component (Q2).** Table 2 provide the contributions of different components of MSJD. The re-

| P | E | A | R | AT | MT | TSLA | | ETH | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | SR↑ | MDD↓ | SR↑ | MDD↓ |
| ✓ | | | | | | 1.33 | 16.52 | 1.57 | 13.51 |
| ✓ | ✓ | | | | | 1.34 | 15.68 | 1.65 | 12.93 |
| ✓ | ✓ | ✓ | | | | 1.63 | 15.68 | 1.77 | 12.36 |
| ✓ | ✓ | ✓ | ✓ | | | 1.84 | 13.47 | 1.80 | 12.07 |
| ✓ | ✓ | ✓ | ✓ | ✓ | | 1.96 | 12.36 | 1.82 | 11.28 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | **2.25** | **11.34** | **2.14** | **9.05** |

Table 2: Ablation studies over different components.

sults reveal that each component plays a crucial role. Specifically, the temporal position encoding(P) establishes a baseline temporal awareness, achieving an SR of 1.33 in the TSLA dataset. This component enables the model to capture time-related features effectively. The introduction of jump diffusion embedding(E) improves the model's ability to decompose time-series data into trend, seasonal, and residual components, resulting in a 1% increase in SR. The jump diffusion attention(A)[1] further enhances the model's responsiveness to market volatility. This component enables the model to dynamically adjust its attention based on temporal dynamics. The routers(R) ensure that the model uses the most appropriate data and processes for each task, contributing to a 13% increase in SR. Finally, the complete framework with Adversarial Post-Training(AT) and Mask Post-Training(MT) methods achieves a 69% improvement in SR over the base model, demonstrating the synergistic integration of all components and their collective impact on improving the model's performance.

| Method | AAPL | AMZN | GOOGL |
|---|---|---|---|
| FinAgent(Stationary) | 1.99 | 1.89 | 2.10 |
| FinAgent(Non-Stationary) | 0.87 | 1.33 | 1.41 |
| **FinAgent(Hybrid)** | **1.43** | **1.61** | **1.78** |
| MSJD(Stationary) | $2.08^{+0.09}$ | $2.13^{+0.24}$ | $2.11^{+0.01}$ |
| MSJD(Non-Stationary) | $1.64^{+0.77}$ | $1.69^{+0.36}$ | $1.83^{+0.42}$ |
| **MSJD(Hybrid)** | $\mathbf{1.86^{+0.43}}$ | $\mathbf{1.91^{+0.30}}$ | $\mathbf{1.97^{+0.19}}$ |

Table 3: Non-Stationary Ablation study.

**Non-Stationary Adaptation(Q3).** We categorize historical data into steady-state and non-stationary data according to whether the variance of the intraday hourly price changes over time, and we perform experiments on each of the two types of data. The sharp results presented in Table 3 highlight the effectiveness of the MSJD framework in adapting to non-stationary market environments. The comparison shows that pure non-stationary methods suffer from a 21% degradation in SR due to overfitting, while stationary approaches fail to perform well during periods of high market volatility. In contrast, the MSJD framework, with its hybrid approach to modeling market dynamics, maintains a consistent SR between 1.86 and 1.97 in different datasets. This demonstrates the framework's ability to effectively handle distribution shifts and adapt to changing market conditions. The hybrid solution, which combines explicit modeling of market dynamics through jump-diffusion processes with implicit encoding in multimodal LLMs, provides a robust and reliable approach to financial price prediction in non-stationary environments. The results underscore the importance of considering both stationary and non-stationary components in modeling financial markets and highlight the advantages of the MSJD framework.

[1]Note that we also modulate the attention weights of $\mathbf{A}$ by a temporal factor $\gamma(t) = (jump/t)$ mechanisms. For details, please refer to the supplementary material.

**Comparison with traditional Stochastic Processes(Q4).** In this section, we compare MSJD with alternative stochastic processes, such as the Geometric Brownian Motion (GBM) (Brătian et al. 2022) and Ornstein-Uhlenbeck (OU) processes (Maller, Müller, and Szimayer 2009) on the TSLA dataset. The results are shown in Table 4.

The results show that MSJD significantly outperforms the GBM and OU-based frameworks in terms of ARR and SR, while maintaining a lower MDD. This demonstrates the superior ability of the Market-State Jump Diffusion process to capture complex market dynamics, including sudden jumps and non-stationary behavior, leading to more accurate and robust predictions.

| Method | ARR↑ | SR↑ | MDD↓ |
|---|---|---|---|
| GBM-based Framework | 78.50 | 1.82 | 15.67 |
| OU-based Framework | 82.30 | 1.94 | 14.21 |
| **MSJD (Proposed)** | $\mathbf{95.14^{+12.84}}$ | $\mathbf{2.25^{+0.31}}$ | $\mathbf{11.34^{-2.87}}$ |

Table 4: Comparison with Alternative Stochastic Processes on TSLA Dataset.

**Impact of Missing Modalities(Q5).** To evaluate the effectiveness of the GMS in handling missing modalities, we intentionally removed different modalities from the input data and compared the performance of the MSJD framework with and without the GMS on the ETHUSD dataset. The results are shown in Table 5.

The results show that the GMS significantly mitigates the impact of missing modalities. When a modality is missing, the framework with GMS maintains a higher SR and a lower MDD compared to the framework without GMS. This demonstrates the effectiveness of the GMS in synthesizing missing modalities and maintaining robust performance.

| Missing Modality | With GMS | | Without GMS | |
|---|---|---|---|---|
| | SR↑ | MDD↓ | SR↑ | MDD↓ |
| Text | **2.14** | **9.05** | 1.82 | 12.54 |
| Image | **2.10** | **9.20** | 1.76 | 13.10 |
| Audio | **2.08** | **9.30** | 1.72 | 13.50 |

Table 5: Impact of Missing Modalities on ETHUSD Dataset.

## Conclusion

We present the Market-State Jump Diffusion Framework (MSJD) for financial price prediction in non-stationary environments. Our solution explicitly models market dynamics

through jump-diffusion processes, while implicitly encoding these patterns in multimodal LLMs. The proposed general modality synthesizer demonstrates 69% performance improvement through adversarial masking training. Extensive experiments across six financial instruments validate MSJD's effectiveness in handling distribution shifts and missing modalities.

## References


Albuquerque, P. H. M.; Peng, Y.; and Silva, J. P. F. d. 2022. Making the whole greater than the sum of its parts: A literature review of ensemble methods for financial time series forecasting. *Journal of Forecasting*, 41(8): 1701–1724.

Anderson, O. D. 1976. Time-Series. 2nd edn.

Avellaneda, M.; and Stoikov, S. 2008. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3): 217–224.

Bertsekas, D. 1996. Neuro-dynamic programming. *Athena Scientific*.

Black, F.; and Scholes, M. 1973. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3): 637–654.

Brătian, V.; Acu, A.-M.; Mihaiu, D. M.; and Șerban, R.-A. 2022. Geometric Brownian Motion (GBM) of stock indexes and financial market uncertainty in the context of non-crisis and financial crisis scenarios. *Mathematics*, 10(3): 309.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chekhlov, A.; Uryasev, S.; and Zabarankin, M. 2005. Drawdown measure in portfolio optimization. *International Journal of Theoretical and Applied Finance*, 8(01): 13–58.

Cho, J. H.; and Hariharan, B. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 4794–4802.

Costabile, M.; Leccadito, A.; Massabó, I.; and Russo, E. 2014. Option pricing under regime-switching jump–diffusion models. *Journal of Computational and Applied Mathematics*, 256: 152–167.

Daley, D. J.; and Vere-Jones, D. 2006. *An introduction to the theory of point processes: volume I: elementary theory and methods*. Springer Science & Business Media.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1555–1564.

Fama, E. F.; and French, K. R. 1992. The cross-section of expected stock returns. *the Journal of Finance*, 47(2): 427–465.

Fama, E. F.; and French, K. R. 1993. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1): 3–56.

Fatemi, S.; and Hu, Y. 2024. FinVision: A Multi-Agent Framework for Stock Market Prediction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, 582–590.

Graves, A.; and Graves, A. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, 37–45.

Gu, S.; Kelly, B.; and Xiu, D. 2020. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5): 2223–2273.

Guéant, O.; Lehalle, C.-A.; and Fernandez-Tapia, J. 2013. Dealing with the inventory risk: a solution to the market making problem. *Mathematics and financial economics*, 7: 477–507.

Hinton, G. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Hochreiter, S. 1997. Long Short-term Memory. *Neural Computation MIT-Press*.

Hyndman, R. J.; and Athanasopoulos, G. 2018. *Forecasting: principles and practice*. OTexts.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*.

Lewellen, J. 2004. Predicting returns with financial ratios. *Journal of Financial Economics*, 74(2): 209–235.

Li, X.; Deng, X.; Zhu, S.; Wang, F.; and Xie, H. 2014. An intelligent market making strategy in algorithmic trading. *Frontiers of Computer Science*, 8: 596–608.

Lin, B.; Tang, Z.; Ye, Y.; Cui, J.; Zhu, B.; Jin, P.; Zhang, J.; Ning, M.; and Yuan, L. 2024. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*.

Maller, R. A.; Müller, G.; and Szimayer, A. 2009. Ornstein–Uhlenbeck processes and extensions. *Handbook of financial time series*, 421–437.

Martinez-Miera, D.; and Repullo, R. 2019. Monetary policy, macroprudential policy, and financial stability. *Annual Review of Economics*, 11(1): 809–832.

Mizrahi, D.; Bachmann, R.; Kar, O.; Yeo, T.; Gao, M.; Dehghan, A.; and Zamir, A. 2023. 4m: Massively multimodal masked modeling. *Advances in Neural Information Processing Systems*, 36: 58363–58408.

Ogasawara, E.; Martinez, L. C.; De Oliveira, D.; Zimbrão, G.; Pappa, G. L.; and Mattoso, M. 2010. Adaptive normalization: A novel data normalization approach for non-stationary time series. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Omi, T.; Aihara, K.; et al. 2019. Fully neural network based model for general temporal point processes. *Advances in neural information processing systems*, 32.

Ozbayoglu, A. M.; Gudelek, M. U.; and Sezer, O. B. 2020. Deep learning for financial applications: A survey. *Applied soft computing*, 93: 106384.

Paparoditis, E. 2010. Validating stationarity assumptions in time series analysis by rolling local periodograms. *Journal of the American Statistical Association*, 105(490): 839–851.

Passalis, N.; Tefas, A.; Kanniainen, J.; Gabbouj, M.; and Iosifidis, A. 2019. Deep adaptive input normalization for time series forecasting. *IEEE transactions on neural networks and learning systems*, 31(9): 3760–3765.

Sawhney, R.; Mathur, P.; Mangal, A.; Khanna, P.; Shah, R. R.; and Zimmermann, R. 2020. Multimodal multi-task financial risk forecasting. In *Proceedings of the 28th ACM international conference on multimedia*, 456–465.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sezer, O. B.; Gudelek, M. U.; and Ozbayoglu, A. M. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90: 106181.

Sharpe, W. F. 1966. Mutual fund performance. *The Journal of business*, 39(1): 119–138.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shchur, O.; Biloš, M.; and Günnemann, S. 2019. Intensity-free learning of temporal point processes. *arXiv preprint arXiv:1909.12127*.

Shumway, R. H.; Stoffer, D. S.; Shumway, R. H.; and Stoffer, D. S. 2017. ARIMA models. *Time series analysis and its applications: with R examples*, 75–163.

Sortino, F. A.; and Price, L. N. 1994. Performance measurement in a downside risk framework. *the Journal of Investing*, 3(3): 59–64.

Sui, M.; Zhang, C.; Zhou, L.; Liao, S.; and Wei, C. 2024. An ensemble approach to stock price prediction using deep learning and time series models. In *2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, 793–797. IEEE.

Vaidya, R. 2020. Moving Average Convergence-Divergence (MACD) Trading Rule: An Application in Nepalese Stock Market" NEPSE". *Quantitative Economics and Management Studies*, 1(6): 366–374.

Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Vyachkileva, D. 2018. Performance Analysis Based on Adequate Risk-Adjusted Performance Measures.

Wu, S.; Fei, H.; Qu, L.; Ji, W.; and Chua, T.-S. 2023. Next-gpt: Any-to-any multimodal llm. *arXiv preprint arXiv:2309.05519*.

Yang, C.; Mei, H.; and Eisner, J. 2021. Transformer embeddings of irregularly spaced events and their participants. *arXiv preprint arXiv:2201.00044*.

Yang, H.; Liu, X.-Y.; and Wang, C. D. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

Yang, L.; Ng, T. L. J.; Smyth, B.; and Dong, R. 2020. Html: Hierarchical transformer-based multi-task learning for volatility prediction. In *Proceedings of The Web Conference 2020*, 441–451.

Yu, Y.; Li, H.; Chen, Z.; Jiang, Y.; Li, Y.; Zhang, D.; Liu, R.; Suchow, J. W.; and Khashanah, K. 2024. FinMem: A performance-enhanced LLM trading agent with layered memory and character design. In *Proceedings of the AAAI Symposium Series*, volume 3, 595–597.

Zhang, D.; Yu, Y.; Dong, J.; Li, C.; Su, D.; Chu, C.; and Yu, D. 2024a. Mm-llms: Recent advances in multimodal large language models. *arXiv preprint arXiv:2401.13601*.

Zhang, S.; Zhou, C.; Liu, Y. A.; Zhang, P.; Lin, X.; and Ma, Z.-M. 2024b. Neural Jump-Diffusion Temporal Point Processes. In *Forty-first International Conference on Machine Learning*.

Zhang, W.; Zhao, L.; Xia, H.; Sun, S.; Sun, J.; Qin, M.; Li, X.; Zhao, Y.; Zhao, Y.; Cai, X.; et al. 2024c. FinAgent: A Multimodal Foundation Agent for Financial Trading: Tool-Augmented, Diversified, and Generalist. *arXiv preprint arXiv:2402.18485*.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer hawkes process. In *International conference on machine learning*, 11692–11702. PMLR.

## Reproducibility Checklist

### 1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) yes

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) yes

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) yes

### 2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) no

If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) no

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) no

2.4. Proofs of all novel claims are included (yes/partial/no) no

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) no

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) no

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) no

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) no

### 3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) yes

If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) yes

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) yes

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) partial

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) yes

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) yes

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) yes

### 4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) yes

If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) yes

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) yes

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) yes

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) yes

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) yes

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) yes

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) yes

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) yes

4.10. This paper states the number of algorithm runs used

to compute each reported result (yes/no) yes

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) yes

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) partial

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) partial