mathematics

MDPI

Article

LévyHyper: A Lévy Process-Driven Dynamic Hypergraph Framework for Stock Return Prediction with Jump-Aware Temporal Modeling

Siyu Luo [1,2] and Junming Chen [1,*]

[1] Faculty of Humanities and Arts, Macau University of Science and Technology, Macau 999078, China; 2023311237@email.cufe.edu.cn
[2] School of International Trade and Economics, Central University of Finance and Economics, Beijing 100081, China
* Correspondence: jmchen@must.edu.mo

Abstract

Stock return prediction for quantitative trading in U.S. equity markets has evolved from parametric econometric modeling toward data-driven deep learning systems that must jointly capture temporal dynamics, discontinuous jumps, and evolving cross-asset dependencies. Existing approaches still face three key challenges in deep learning-based stock return prediction: jump-aware temporal modeling is often missing or handled by ad hoc heuristics; higher-order stock relations are frequently encoded by static graphs/hypergraphs that do not adapt across market conditions, and temporal and relational learning are commonly implemented as sequential blocks with limited bidirectional interaction. We propose LévyHyper, an end-to-end framework that unifies jump-aware temporal encoding with regime-adaptive dynamic hypergraph learning and multi-scale hypergraph reasoning. LévyHyper integrates a neural jump-aware temporal layer motivated by Lévy jump-diffusion modeling, a regime-weighted fusion of predefined and learned hyperedges via a differentiable constructor, and a multi-scale hypergraph convolution module for hierarchical temporal aggregation. Experiments on S&P 500 data (463 stocks, 10 evaluation phases, prediction horizon $\tau = 5$ trading days) show that LévyHyper improves IC/RankIC and portfolio-level Sharpe ratio over strong baselines on average. We additionally report uncertainty estimates, significance tests, and transaction-cost sensitivity to support robust conclusions.

Keywords: Lévy process; dynamic hypergraph learning; stock return prediction; jump-diffusion models; financial time series

MSC: 00A71; 05C85; 05C50

Check for updates

Academic Editor: Abolfazl Saghafi

Received: 19 January 2026
Revised: 12 February 2026
Accepted: 15 February 2026
Published: 17 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The global stock market, with a total capitalization exceeding US$111 trillion in 2024, has become a highly coupled system where asset returns reflect both complex temporal dependencies and evolving cross-asset interactions. Accurate return prediction is central to quantitative trading, risk management, and portfolio construction. This shift has been accelerated by increased data availability and the success of deep learning models, alongside the recognition that stock movements are not independent but shaped by multi-asset dependencies and shared market factors.

Real-world markets exhibit discontinuous return dynamics with heavy tails and clustered extremes; time-varying group behaviors (e.g., sector rotations and crisis-driven correlation breakdowns), and predictive signals that manifest at multiple horizons (daily/weekly) and interact with relational structure. These properties motivate models that can jointly encode jump-aware temporal patterns and adaptive higher-order relationships.

From a methodological perspective, stock forecasting has progressed through three paradigms. Classical econometric models such as ARCH [1] and GARCH [2] provide tractable volatility dynamics under parametric assumptions, but they often struggle with nonlinear dependencies and distributional misspecification. Realized volatility measures [3] and HAR [4] improved practical accuracy via multi-horizon components, yet they remain limited in handling abrupt regime shifts and cross-asset spillovers. More recently, deep learning approaches [5–7] have shown strong capacity for learning nonlinear patterns, where transformers [6,8–10] and graph-based methods [6,11,12] achieve competitive results by modeling temporal structure and inter-stock relations.

Despite these advances, important gaps remain. The challenges are shown in Figure 1. First, there is jump-aware temporal modeling under heavy-tailed dynamics. Financial returns exhibit abrupt jumps that deviate from continuous diffusion assumptions, with a non-trivial fraction of variance attributable to jump components [13]. Many neural forecasting pipelines either ignore jump behavior or rely on heuristic detection [3]. Self-exciting processes such as Hawkes [14] model event clustering, but practical forecasting still requires mechanisms that flexibly represent time-varying jump activity and heavy-tailed jump sizes under changing market conditions. While self-exciting processes such as Hawkes models can capture temporal clustering of jumps, they typically assume stationary baseline intensities and fixed parametric forms, limiting adaptation to time-varying market conditions. Furthermore, existing Hawkes-based forecasting models operate primarily in the temporal domain without integrating jump dynamics with evolving cross-asset relational structures. LévyHyper addresses these limitations through: (1) Neural parameterization of time-varying Lévy components that adapt to market states, (2) End-to-end integration of jump-aware temporal encoding with dynamic hypergraph learning where jump intensities and relational structures co-evolve, (3) Multi-scale spectral convolution propagating jump information across stocks through evolving higher-order relationships.

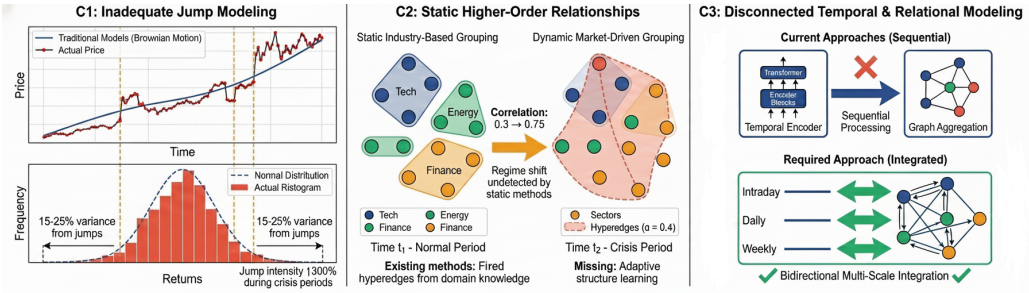


Figure 1. Three challenges in deep learning–based stock forecasting: (**C1**) discontinuous price dynamics with heavy–tailed jumps; (**C2**) static hypergraph structures that cannot adapt to regime shifts; (**C3**) weak coupling between multi–scale temporal modeling and relational learning due to sequential architectures.

Second, higher-order relations are dynamic, but hypergraphs are often static. Beyond pairwise correlations, stocks form group-wise dependencies that evolve over time. Hypergraph approaches [11,15] move beyond pairwise edges, but they commonly rely on static hyperedges derived from domain knowledge (e.g., industry sectors). Such structures cannot fully reflect regime-dependent relationship shifts; for instance, crisis-like periods

can sharply change cross-sector coupling [16]. Dynamic graph learning methods [17] often update pairwise weights rather than evolving hyperedge structure, limiting their ability to capture emergent group behaviors.

Third, Multi-scale temporal dynamics and relational learning are weakly coupled. Effective return prediction requires modeling signals across multiple horizons while leveraging cross-asset structure. Transformers [8,18] capture long-range temporal dependencies but often encode each stock largely independently during temporal representation learning. Graph/hypergraph methods [6,11,19] capture cross-asset information but may rely on simpler temporal aggregation. Cascaded designs (temporal then relational) [15] can limit mutual reinforcement between temporal and relational representations [20], and many methods operate at a single temporal resolution despite the hierarchical nature of market dynamics.

To address these challenges, we propose LévyHyper, an end-to-end framework that unifies jump-aware temporal encoding with regime-adaptive dynamic hypergraph learning for stock return prediction and cross-sectional ranking. Specifically, LévyHyper combines (i) a Lévy jump-diffusion motivated temporal module to better represent discontinuities and heavy tails, (ii) a regime-conditioned dynamic hypergraph constructor that fuses predefined hyperedges with learned hyperedges, and (iii) a multi-scale spectral hypergraph reasoning module to capture hierarchical structure across temporal granularities. We note that individual ingredients connect to existing lines of work (e.g., neural point processes for event intensities, dynamic graph learning, and spectral/wavelet message passing). Our goal is not to claim each component as a standalone breakthrough, but to provide a principled and practical unification that enables jump-aware representations, evolving higher-order structure, and multi-scale relational reasoning within one differentiable pipeline under a consistent phase-based protocol.

The main contributions of this work are summarized as follows:

(1) We propose a neural Lévy jump-diffusion layer that models discontinuous price movements through compound Poisson processes with learnable time-varying intensities. Our formulation employs neural networks to parameterize jump dynamics conditioned on market states, enabling adaptive modeling across volatility regimes. We provide theoretical analysis showing approximation guarantees and derive closed-form conditional intensity functions for efficient optimization.

(2) We introduce a regime-adaptive dynamic hypergraph framework that fuses domain knowledge with data-driven discovery of latent market relationships. Our approach combines predefined industry-based hyperedges with dynamically generated structures learned through attention mechanisms, formulated as a differentiable top-k selection problem via Gumbel-Softmax relaxation for end-to-end training.

(3) We develop a multi-scale spectral hypergraph convolution operator that performs joint temporal-relational learning through wavelet-based decomposition. Our formulation extends spectral graph convolution to hypergraphs with multi-resolution temporal analysis, proven to satisfy permutation equivariance, spectral localization, and linear computational complexity.

(4) We conduct comprehensive experiments on S&P 500 market data with 463 stocks across 10 phases spanning 2010–2019. Ablation studies and portfolio backtesting confirm the contribution of each component under diverse market conditions within the U.S. equity market setting.

The remainder of this paper is organized as follows: Section 2 reviews related works. Section 3 presents our proposed LévyHyper framework. Section 4 describes the experimental setup and evaluation results. Section 5 concludes this paper with discussions and future research directions.

2. Related Work

This work builds upon and extends three interconnected research streams in financial forecasting: deep temporal modeling for stock prediction, graph and hypergraph neural networks for capturing inter-stock dependencies, and integrated approaches that combine temporal and relational learning. While each stream has made significant progress independently, substantial gaps remain in modeling discontinuous price dynamics, adaptive higher-order relationships, and bidirectional temporal-relational coupling. Our framework addresses these limitations through a principled unification of Lévy process theory with dynamic hypergraph learning.

2.1. Deep Learning for Stock Forecasting

The evolution of deep learning methods for stock forecasting has progressed through three distinct phases, beginning with recurrent architectures and advancing toward attention-based mechanisms. Early approaches applied recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks [21] to capture temporal dependencies in price and return sequences. Chen et al. [5] demonstrated that LSTM-based models could outperform traditional econometric methods by learning complex nonlinear patterns from raw market data. However, these recurrent architectures suffered from fundamental limitations: gradient vanishing problems during backpropagation through time, difficulty capturing long-range dependencies spanning weeks or months, and computational inefficiency due to sequential processing constraints.

The introduction of attention mechanisms and transformer architectures [22] marked a paradigm shift by enabling flexible dependency learning without recurrent bottlenecks. Ding et al. [8] proposed HMG-TF (Hierarchical Multi-scale Graph Transformer), which captures hierarchical temporal structure through multi-resolution attention across daily, weekly, and monthly patterns. Li et al. [18] developed MASTER, a sequential attention framework that stacks temporal, stock-wise, and temporal attention layers to capture cross-time correlations. These transformer-based methods achieve strong performance on standard benchmarks and demonstrate superior capacity for modeling long-horizon dependencies compared to recurrent predecessors. Classical econometric models such as ARCH and GARCH provide tractable volatility dynamics under parametric assumptions, but they often struggle with nonlinear dependencies and distributional misspecification. Realized volatility measures and HAR improved practical accuracy via multi-horizon components, yet they remain limited in handling abrupt regime shifts and cross-asset spillovers. Prior work has demonstrated that these traditional methods are consistently outperformed by deep learning approaches on standard benchmarks, which motivates our focus on neural network-based baselines that represent the current state-of-the-art.

Despite these advances, most temporal modeling approaches exhibit a critical limitation: they process each stock independently during temporal encoding, failing to leverage cross-asset information that could enhance predictive accuracy. Furthermore, existing methods predominantly assume continuous price dynamics governed by Brownian motion, neglecting the discontinuous jumps and heavy-tailed distributions that characterize real financial markets. Our Lévy jump-diffusion temporal encoder addresses both gaps by explicitly modeling jump dynamics through compound Poisson processes while integrating cross-asset relational signals during temporal representation learning.

2.2. Graph and Hypergraph Neural Networks for Financial Markets

Graph Neural Networks (GNNs) [23] have emerged as powerful tools for modeling inter-stock dependency structures by treating stocks as nodes and relationships as edges. Feng et al. [6] proposed RSR (Relational Stock Ranking), which constructs stock

relation graphs using industry classifications and web-crawled metadata, allowing edge weights to evolve over time through learned attention mechanisms. Matsunaga et al. [24] explored temporal graph convolutional networks that jointly model price dynamics and correlation structures by propagating information across both temporal and relational dimensions. Zheng et al. [17] introduced RT-GCN (Relational Temporal Graph Convolutional Network), constructing three-dimensional graphs that incorporate temporal, industry-based, and correlation-based relationships from multiple data sources. These graph-based approaches demonstrate that explicitly modeling inter-stock dependencies improves forecasting accuracy compared to treating stocks as independent entities.

However, traditional pairwise graphs are fundamentally limited in their ability to represent higher-order group interactions that frequently occur in financial markets. For instance, sector-wide momentum effects, regulatory policy impacts on industry clusters, and contagion dynamics during market crises all involve simultaneous interactions among multiple stocks that cannot be adequately captured through pairwise edges alone. Hypergraph neural networks address this limitation by introducing hyperedges—structural elements that connect arbitrary subsets of nodes simultaneously. Sawhney et al. [11] proposed STHAN-SR (Spatio-Temporal Hypergraph Attention Network for Stock Ranking), constructing hypergraphs from industry classifications and Wikidata knowledge graphs, then applying hypergraph convolution with attention mechanisms to model group-wise dependencies. Huynh et al. [15] developed ESTIMATE (Efficient Spatio-Temporal Modeling with Adaptive Hypergraph Attention), which augments industry-based hypergraphs with correlation-based hyperedges and employs wavelet hypergraph convolution to achieve computational efficiency while preserving spectral properties.

While these hypergraph-based methods validate the value of higher-order relational modeling and achieve state-of-the-art performance on multiple benchmarks, they predominantly rely on static or weakly adaptive hypergraph structures. Predefined hyperedges based on industry sectors or fixed correlation thresholds cannot capture the dynamic nature of market regimes, where relationship patterns shift substantially during transitions between normal and crisis periods. Our regime-adaptive dynamic hypergraph constructor addresses this limitation by learning time-varying hyperedge structures through attention-based generation mechanisms, enabling the model to discover latent group relationships that emerge during different market conditions while preserving interpretable domain knowledge through weighted fusion.

2.3. Coupling Temporal and Relational Learning

A fundamental challenge in financial forecasting lies in effectively coupling temporal encoders with relational aggregation mechanisms such that both representations mutually reinforce each other rather than operating in isolation. Many existing pipelines adopt cascaded architectures where temporal features are first extracted through RNNs or transformers and subsequently passed to graph or hypergraph propagation layers for relational aggregation [11,15]. While this sequential approach achieves reasonable performance and maintains architectural simplicity, it imposes a critical restriction: information flows unidirectionally from temporal modeling to relational aggregation, preventing feedback from relational learning from influencing temporal representation construction. This limitation becomes particularly problematic when relational structures themselves evolve with market conditions, as the temporal encoder cannot adapt its representations based on changing group dynamics.

Recent work has begun exploring bidirectional coupling strategies, though with limited scope. Some methods alternate between temporal and graph convolution layers [17], allowing limited interaction through layer stacking but still processing temporal and re-

lational information at separate stages. Others employ joint attention mechanisms that compute similarity scores across both time and stock dimensions [25], but these approaches typically operate at a single temporal resolution and use pairwise rather than higher-order relationships. The absence of unified frameworks that enable true bidirectional information flow between multi-scale temporal dynamics and evolving higher-order relational structures represents a fundamental gap in current methodologies.

Our multi-scale spectral hypergraph convolution module addresses this limitation through simultaneous joint learning across temporal scales and relational structures. By performing wavelet-based decomposition on hypergraph-structured data, our approach enables temporal patterns at different resolutions (daily fluctuations, weekly trends, monthly momentum) to interact with dynamically evolving group-wise dependencies. This design ensures that relational information influences temporal feature extraction while temporal patterns guide hypergraph structure adaptation, creating a feedback loop that enhances both representations. Furthermore, our framework integrates discontinuous jump dynamics through the Lévy layer, allowing the model to capture regime-dependent relationships where jump intensities and hypergraph structures co-evolve during market transitions. This principled unification of temporal modeling, relational learning, and stochastic process theory distinguishes our approach from prior cascaded or weakly coupled architectures.

3. Methodology

3.1. Problem Formulation and Notation

We begin by formally defining the stock volatility forecasting problem and establishing the mathematical notation used throughout this paper.

Let $S = \{1, 2, \ldots, S\}$ denote a universe of S stocks. For each stock $s \in S$, we observe high-frequency price data over a historical window $T = \{1, 2, \ldots, T\}$. At each time step $t \in T$, the feature vector $x_t^s \in R^F$ encodes F features including OHLCV (Open, High, Low, Close, Volume) ratios and technical indicators. The historical data tensor is denoted as $X \in R^{S \times T \times F}$, where $X_{s,t,:} = x_t^s$.

The target variable is the return ratio $r_{T+\tau}^s$ for stock s at lookahead time $T + \tau$, defined as:

$$r_{T+\tau}^s = \frac{c_{T+\tau}^s - c_T^s}{c_T^s}, \tag{1}$$

where c_t^s denotes the closing price of stock s at time t, and τ is the prediction horizon.

The objective is to learn a function $f_\theta : R^{S \times T \times F} \to R^S$ parameterized by θ that minimizes the expected prediction error:

$$\theta^* = \arg\min_\theta E[L(r_{T+\tau}, f_\theta(X))], \tag{2}$$

where $r_{T+\tau} = [r_{T+\tau}^1, \ldots, r_{T+\tau}^S]^\top \in R^S$ is the vector of true return ratios, and $L(\cdot, \cdot)$ is a loss function (e.g., mean absolute error or mean squared error).

3.2. Framework Overview

The LévyHyper architecture consists of four main components that operate in a sequential yet interconnected manner:

Feature Embedding and Positional Encoding. Raw input features X are first embedded into a higher-dimensional latent space through learnable linear transformations, with sinusoidal positional encodings added to preserve temporal ordering information for subsequent attention mechanisms; Lévy Jump-Diffusion Temporal Encoder. The embedded features are processed through a multi-layer temporal encoder that combines transformer-style self-attention with our novel Lévy jump-diffusion layer. This component

captures both continuous price evolution and discontinuous jump dynamics, producing enriched temporal representations $H \in R^{S\times T\times d}$ where d is the hidden dimension; Regime-Adaptive Dynamic Hypergraph Constructor. Based on the temporal representations, this module dynamically generates hypergraph structures $G_t = (V, E_t, W_t)$ that evolve over time. The hyperedge set E_t combines predefined domain knowledge (industry sectors, correlation-based groups) with data-driven hyperedges learned through attention mechanisms, while hyperedge weights W_t are determined by a regime-detection network; Multi-Scale Spectral Hypergraph Convolution. The temporal representations and dynamic hypergraphs are jointly processed through wavelet-based multi-resolution hypergraph convolutions. This module performs message passing across hyperedges at multiple time scales (daily, weekly, monthly), producing final stock representations $Z \in R^{S\times d}$ that encode both temporal dynamics and relational structures. The final predictions are obtained through a fully connected output layer: $\hat{r}_{T+\tau} = W^{out}Z + b^{out}$, where $W^{out} \in R^{S\times d}$ and $b^{out} \in R^{S}$ are learnable parameters. The framework is shown in Figure 2.

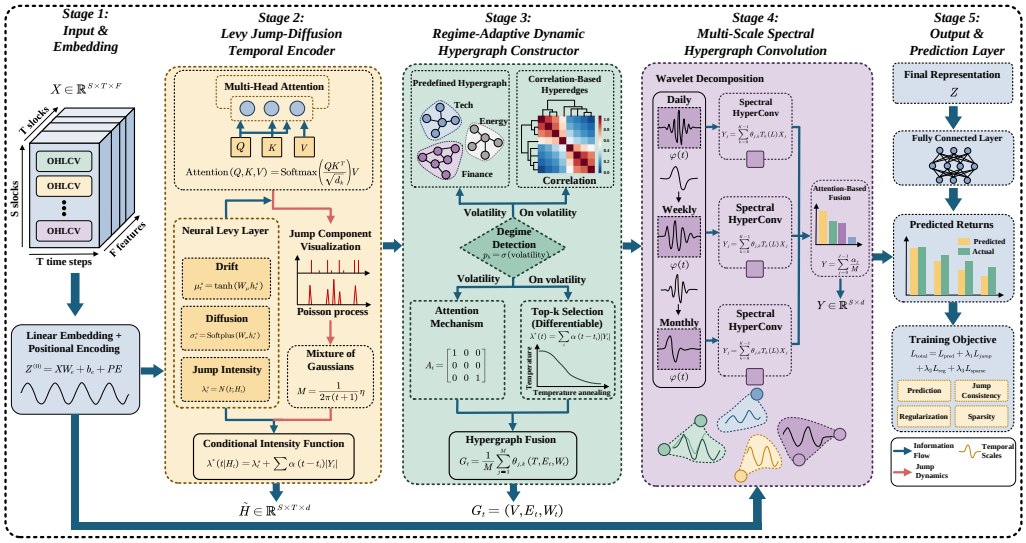


Figure 2. Architecture of LévyHyper framework. The model integrates five key components: (1) feature embedding with positional encoding, (2) Lévy jump-diffusion temporal encoder with neural-parameterized jump dynamics, (3) regime-adaptive dynamic hypergraph constructor fusing domain knowledge with learned structures, (4) multi-scale spectral hypergraph convolution across daily/weekly/monthly resolutions, and (5) prediction layer with composite loss function. Information flows from raw OHLCV data through temporal and relational modeling to final return predictions.

3.3. Lévy Jump-Diffusion Temporal Encoder

3.3.1. Theoretical Foundation of Lévy Processes

We begin by establishing the theoretical foundation for modeling stock price dynamics using Lévy processes, which provide a rigorous framework for capturing both continuous diffusion and discontinuous jumps.

A stochastic process $\{L_t\}_{t\geq 0}$ defined on a probability space (Ω, F, P) is called a Lévy process if it satisfies: $L_0 = 0$ almost surely, Independent increments: For any $0 \leq t_1 < t_2 < \cdots < t_n$, the random variables $L_{t_2} - L_{t_1}, \ldots, L_{t_n} - L_{t_{n-1}}$ are independent, Stationary increments: For any $s, t \geq 0$, $L_{t+s} - L_s$ has the same distribution as L_t, Stochastic continuity: For any $\epsilon > 0$ and $t \geq 0$, $\lim_{s\to t} P(|L_s - L_t| > \epsilon) = 0$.

The Lévy–Khintchine representation theorem [26] provides a complete characterization of Lévy processes through their characteristic function: For any Lévy process

$\{L_t\}_{t\geq 0}$, there exist constants $\mu \in R$, $\sigma^2 \geq 0$, and a Lévy measure ν on $R \setminus \{0\}$ satisfying $\int_R \min(1, x^2)\nu(dx) < \infty$ such that:

$$E[e^{iuL_t}] = \exp(t\psi(u)), \tag{3}$$

where the characteristic exponent $\psi(u)$ is given by:

$$\psi(u) = i\mu u - \frac{1}{2}\sigma^2 u^2 + \int_{R\setminus\{0\}} \left(e^{iux} - 1 - iux1_{|x|<1}\right)\nu(dx). \tag{4}$$

The triplet (μ, σ^2, ν) uniquely determines the Lévy process. The drift parameter μ represents the deterministic trend, σ^2 governs the continuous Brownian component, and the Lévy measure ν characterizes the jump structure. For stock price modeling, we focus on compound Poisson processes, a special class of Lévy processes where jumps occur at random times governed by a Poisson process.

A compound Poisson process $\{X_t\}_{t\geq 0}$ is defined as:

$$X_t = \sum_{i=1}^{N_t} Y_i, \tag{5}$$

where $\{N_t\}_{t\geq 0}$ is a Poisson process with intensity $\lambda > 0$, and $\{Y_i\}_{i=1}^{\infty}$ are i.i.d. random variables representing jump sizes, independent of N_t.

For financial applications, we extend this to a jump-diffusion model:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + S_{t-} dX_t, \tag{6}$$

where S_t is the stock price, W_t is a standard Brownian motion, and X_t is a compound Poisson process independent of W_t. The Lévy–Itô decomposition provides a canonical representation of any Lévy process as the sum of three independent components: a linear drift term, a Brownian motion with constant volatility, and a pure jump process. Formally, any Lévy process $\{L_t\}_{t\geq 0}$ can be decomposed as $L_t = \mu t + \sigma W_t + X_t$, where $\mu \in R$ is the drift coefficient, $\sigma \geq 0$ is the diffusion coefficient, W_t is a standard Brownian motion, and X_t is a pure jump process characterized by the Lévy measure ν. This decomposition is unique and provides the foundation for our neural parameterization.

3.3.2. Neural Parameterization of Time-Varying Lévy Dynamics

A key limitation of classical Lévy models is their assumption of constant parameters (μ, σ, λ), which fails to capture the time-varying nature of financial markets. We address this by introducing neural network parameterizations that allow these quantities to adapt based on historical market states.

Let $h_t^s \in R^d$ denote the hidden state representation for stock s at time t, obtained from the temporal encoder (details in Section 3.3.4). We parameterize the time-varying Lévy process components as:

$$\begin{aligned} \mu_t^s &= \text{Tanh}(W_\mu h_t^s + b_\mu), \\ \sigma_t^s &= \text{Softplus}(W_\sigma h_t^s + b_\sigma), \\ \lambda_t^s &= \text{Softplus}(W_\lambda h_t^s + b_\lambda), \end{aligned} \tag{7}$$

where $W_\mu, W_\sigma, W_\lambda \in R^{1\times d}$ and $b_\mu, b_\sigma, b_\lambda \in R$ are learnable parameters. The Tanh activation ensures $\mu_t^s \in [-1, 1]$, while Softplus guarantees positivity of σ_t^s and λ_t^s.

For the jump size distribution, we employ a mixture of Gaussians to capture both small and large jumps:

$$Y_i \sim \sum_{k=1}^{K} \pi_k N(\mu_k, \sigma_k^2), \tag{8}$$

where K is the number of mixture components, and $\{\pi_k, \mu_k, \sigma_k\}_{k=1}^{K}$ are learned through maximum likelihood estimation on historical jump data.

3.3.3. Conditional Intensity Function and Neural Point Process Layer

To enable efficient gradient-based learning, we derive the conditional intensity function of our time-varying Lévy process. The intensity function $\lambda^*(t|H_t)$ represents the instantaneous expected rate of jump occurrences conditioned on the history $H_t = \{(t_i, Y_i) : t_i < t\}$.

For the neural parameterized jump-diffusion model in Equation (6) with time-varying intensity λ_t^s given by Equation (7), the conditional intensity function is:

$$\lambda^*(t|H_t) = \lambda_t^s + \sum_{i:t_i<t} \alpha(t - t_i)|Y_i|, \tag{9}$$

where $\alpha(t - t_i) = \beta \exp(-\gamma(t - t_i))$ is a decay kernel with learnable parameters $\beta, \gamma > 0$, and $|Y_i|$ is the absolute jump size.

The conditional intensity is derived from the compensator of the counting process N_t. For a time-inhomogeneous Poisson process with rate λ_t, the compensator is $\Lambda_t = \int_0^t \lambda_s ds$. The self-exciting component arises from the feedback effect where large jumps increase the probability of subsequent jumps. Through Doob–Meyer decomposition, we have:

$$N_t = \Lambda_t + M_t, \tag{10}$$

where M_t is a martingale. The conditional intensity is then:

$$\lambda^*(t|H_t) = \frac{d\Lambda_t}{dt} = \lambda_t + \sum_{i:t_i<t} \alpha(t - t_i)|Y_i|, \tag{11}$$

where the second term captures the self-exciting effect with exponential decay.

We implement this as a neural layer that computes the intensity-weighted temporal representation:

$$\tilde{h}_t^s = h_t^s \odot \sigma(\lambda^*(t|H_t)) + W_j \sum_{i:t_i<t} \exp(-\gamma(t - t_i))y_i, \tag{12}$$

where $\odot$ denotes element-wise multiplication, $\sigma(\cdot)$ is the sigmoid function, $W_j \in R^{d\times d}$ is a learnable weight matrix, and $y_i \in R^d$ is the embedded jump feature at time t_i.

3.3.4. Multi-Head Temporal Self-Attention with Positional Encoding

Before applying the Lévy layer, we first extract temporal representations using multi-head self-attention. The input features $X \in R^{S\times T\times F}$ are first embedded:

$$Z^{(0)} = XW_e + b_e + \text{PE}, \tag{13}$$

where $W_e \in R^{F\times d}$, $b_e \in R^d$, and PE denotes sinusoidal positional encoding:

$$\text{PE}_{t,2i} = \sin(t/10000^{2i/d}), \quad \text{PE}_{t,2i+1} = \cos(t/10000^{2i/d}). \tag{14}$$

For each stock s, we apply multi-head self-attention over the temporal dimension:

$$\begin{aligned} Q^s = Z^{(0)}_{s,:,:} W_Q, \quad K^s = Z^{(0)}_{s,:,:} W_K, \quad V^s = Z^{(0)}_{s,:,:} W_V, \\ \text{Attention}(Q^s, K^s, V^s) = \text{Softmax}\left(\frac{Q^s (K^s)^\top}{\sqrt{d_k}} + M \right) V^s, \end{aligned} \tag{15}$$

where $W_Q, W_K, W_V \in R^{d \times d_k}$ are learnable projection matrices, and $M \in R^{T \times T}$ is a causal mask ensuring that attention at time t only depends on times $\leq t$:

$$M_{ij} = \begin{cases} 0, & \text{if } i \geq j, \\ -\infty, & \text{if } i < j. \end{cases} \tag{16}$$

The multi-head attention concatenates H attention heads:

$$\text{MultiHead}(Z^{(0)}_{s,:,:}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H) W_O, \tag{17}$$

where $\text{head}_h = \text{Attention}(Q^s_h, K^s_h, V^s_h)$ and $W_O \in R^{Hd_k \times d}$.

A feed-forward network with residual connection follows:

$$H^s = \text{LayerNorm}\Big(\text{FFN}(\text{MultiHead}(Z^{(0)}_{s,:,:})) + Z^{(0)}_{s,:,:} \Big), \tag{18}$$

where $\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$.

3.3.5. Integrating Lévy Dynamics into Temporal Representations

The Lévy layer (Equation (12)) is applied after the temporal attention to enrich representations with jump dynamics:

$$H = \{ \tilde{H}^s = \text{LévyLayer}(H^s) : s \in S \}, \tag{19}$$

where $H \in R^{S \times T \times d}$ contains the jump-enhanced temporal representations.

Under mild regularity conditions on the true jump-diffusion process, the neural Lévy layer with sufficient hidden dimensions can approximate any Lévy process with bounded variation to arbitrary accuracy.

The proposed jump-aware temporal layer is designed to flexibly represent a broad class of time-varying jump-diffusion dynamics encountered in practice. Intuitively, neural parameterizations can model nonlinear, time-varying drift/volatility terms, while a mixture-based jump-size model can fit multi-modal and heavy-tailed jump patterns given sufficient capacity and appropriate regularization. We treat this as an expressivity motivation rather than a universal approximation guarantee.

3.4. Regime-Adaptive Dynamic Hypergraph Constructor

3.4.1. Hypergraph Preliminaries and Notation

We begin by formally defining the hypergraph structure used in our framework.

A hypergraph is a tuple $G = (V, E, W)$, where: $V = \{v_1, \ldots, v_S\}$ is the set of nodes (stocks), $E = \{e_1, \ldots, e_M\}$ is the set of hyperedges, where each $e_j \subseteq V$ can connect any number of nodes, $W = \text{diag}(w_1, \ldots, w_M) \in R^{M \times M}$ is a diagonal matrix of hyperedge weights.

The structure is represented by an incidence matrix $H \in \{0, 1\}^{S \times M}$:

$$H_{ij} = \begin{cases} 1, & \text{if } v_i \in e_j, \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

We define the node degree matrix $D_v = \text{diag}(d_1, \ldots, d_S)$ where $d_i = \sum_{j=1}^{M} w_j H_{ij}$, and the hyperedge degree matrix $D_e = \text{diag}(\delta_1, \ldots, \delta_M)$ where $\delta_j = \sum_{i=1}^{S} H_{ij}$.

3.4.2. Predefined Hypergraph Construction from Domain Knowledge

Following Sawhney et al. [11] and Huynh et al. [15], we construct a predefined hypergraph $G^{pred} = (V, E^{pred}, W^{pred})$ that encodes domain knowledge about stock relationships. The predefined hyperedge set consists of two components:

Let $I = \{I_1, \ldots, I_K\}$ denote a partition of stocks into K industry sectors (e.g., Technology, Energy, Finance). Each industry I_k forms a hyperedge:

$$E^{ind} = \{e_k^{ind} = \{s : s \in I_k\} : k = 1, \ldots, K\}. \tag{21}$$

We compute pairwise correlations ρ_{ij} between stocks i and j using historical returns over a rolling window. Stocks with high correlations are grouped into hyperedges using hierarchical clustering:

$$E^{corr} = \{e_1^{corr}, \ldots, e_L^{corr}\}, \tag{22}$$

where each e_ℓ^{corr} contains stocks in the ℓ-th cluster with average pairwise correlation exceeding a threshold ρ_{min}.

The predefined hyperedge set is $E^{pred} = E^{ind} \cup E^{corr}$, with uniform weights $W^{pred} = I$.

3.4.3. Dynamic Hyperedge Generation via Attention Mechanism

While predefined hypergraphs capture known relationships, they cannot adapt to evolving market regimes. We introduce a data-driven mechanism to dynamically generate hyperedges based on learned stock embeddings.

At each time step t, we aggregate the temporal representations to obtain stock embeddings:

$$z_t^s = \frac{1}{T} \sum_{\tau=1}^{T} \tilde{h}_\tau^s \in R^d. \tag{23}$$

We then compute pairwise attention scores between all stocks:

$$A_t = \text{Softmax}\left(\frac{Z_t W_q (Z_t W_k)^\top}{\sqrt{d}} \right) \in R^{S \times S}, \tag{24}$$

where $Z_t = [z_t^1; \ldots; z_t^S] \in R^{S \times d}$ and $W_q, W_k \in R^{d \times d}$ are learnable projection matrices.

To form hyperedges, we need to select groups of stocks with high mutual attention. This is formulated as a top-k selection problem:

$$E_t^{dyn} = \left\{ e_j^{dyn} = \{s : A_t[s, c_j] \geq \tau_j\} : j = 1, \ldots, N_{dyn} \right\}, \tag{25}$$

where c_j is the j-th cluster center (initialized randomly and updated during training), τ_j is a threshold, and N_{dyn} is the number of dynamic hyperedges.

However, the top-k operation is non-differentiable. We address this using the Gumbel-Softmax trick [27]:

$$\tilde{H}_{ij}^{dyn} = \frac{\exp((\log A_t[i, c_j] + g_{ij})/\tau_{gs})}{\sum_{i'=1}^{S} \exp((\log A_t[i', c_j] + g_{i'j})/\tau_{gs})}, \tag{26}$$

where $g_{ij} \sim \text{Gumbel}(0, 1)$ are i.i.d. Gumbel noise samples, and τ_{gs} is the temperature parameter (annealed during training).

This produces a soft incidence matrix $\tilde{H}^{dyn} \in [0, 1]^{S \times N_{dyn}}$ that approximates the discrete hypergraph structure while remaining differentiable.

3.4.4. Regime Detection and Hypergraph Fusion

To adaptively weight the predefined and dynamic hypergraphs conditioned on observed market features, we introduce a lightweight gating network that outputs a latent mixing weight. For clarity, we refer to the two ends of this continuum as high-volatility-like and low-volatility-like conditions, without implying causal identification of market regimes.

Let $v_t = \frac{1}{S}\sum_{s=1}^{S} \mathrm{Var}(h^s_{t-w:t})$ be the average variance of stock representations over a window w. The regime probability is computed as:

$$p_t^{high} = \sigma(w_r^\top [v_t; z_t^{global}] + b_r), \tag{27}$$

where $z_t^{global} = \frac{1}{S}\sum_{s=1}^{S} z_t^s$ is the global market embedding, $w_r \in R^{2d}$ and $b_r \in R$ are learnable parameters.

The final hypergraph is a weighted combination:

$$\begin{aligned} H_t &= [(1-p_t^{high})H^{pred}, p_t^{high}\tilde{H}_t^{dyn}] \in R^{S\times(M+N_{dyn})}, \\ W_t &= \mathrm{diag}((1-p_t^{high})1_M, p_t^{high}1_{N_{dyn}}) \in R^{(M+N_{dyn})\times(M+N_{dyn})}, \end{aligned} \tag{28}$$

where 1_M denotes an M-dimensional vector of ones.

The regime-adaptive hypergraph $G_t = (V, E_t, W_t)$ satisfies the following: As p_t^{high} varies smoothly, the hypergraph structure changes continuously, ensuring stable gradient flow during training; In low-volatility regimes ($p_t^{high} \approx 0$), the hypergraph relies primarily on domain knowledge, while in high-volatility regimes ($p_t^{high} \approx 1$), data-driven hyperedges dominate; The combined hypergraph can represent any convex combination of predefined and dynamic structures.

3.5. Multi-Scale Spectral Hypergraph Convolution

3.5.1. Spectral Hypergraph Convolution Operator

We extend spectral graph convolution [28] to the hypergraph domain. The normalized hypergraph Laplacian is defined as:

$$L = I - D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2}, \tag{29}$$

where D_v and D_e are the node and hyperedge degree matrices.

The eigendecomposition of L is $L = U\Lambda U^\top$, where $U = [u_1, \ldots, u_S]$ contains the eigenvectors and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_S)$ contains the eigenvalues.

A spectral filter $g_\theta(\Lambda)$ is applied in the spectral domain:

$$Y = U g_\theta(\Lambda) U^\top X, \tag{30}$$

where $X \in R^{S\times d}$ is the input node feature matrix.

To ensure computational efficiency and spatial localization, we use Chebyshev polynomial approximation [29]:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \tag{31}$$

where T_k is the k-th Chebyshev polynomial, $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I$ is the rescaled eigenvalue matrix, and $\{\theta_k\}_{k=0}^{K-1}$ are learnable parameters.

The hypergraph convolution operation becomes:

$$Y = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) X, \tag{32}$$

where $\tilde{L} = 2L/\lambda_{max} - I$ and $T_k(\tilde{L})$ is computed recursively:

$$T_0(\tilde{L}) = I, \quad T_1(\tilde{L}) = \tilde{L}, \quad T_k(\tilde{L}) = 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L}). \tag{33}$$

3.5.2. Wavelet-Based Multi-Resolution Analysis

To capture temporal dynamics at multiple scales (daily, weekly, monthly), we incorporate wavelet transforms into the hypergraph convolution. Wavelets provide a natural framework for multi-resolution analysis by decomposing signals into components at different frequency bands.

We use the Haar wavelet, defined by the scaling function $\phi(t)$ and wavelet function $\psi(t)$:

$$\phi(t) = \begin{cases} 1, & 0 \le t < 1, \\ 0, & \text{otherwise}, \end{cases} \quad \psi(t) = \begin{cases} 1, & 0 \le t < 0.5, \\ -1, & 0.5 \le t < 1, \\ 0, & \text{otherwise}. \end{cases} \tag{34}$$

For a temporal signal $h_t^s \in R^d$ at time t, we compute wavelet coefficients at scale j:

$$\begin{aligned} a_j^s &= \sum_{t \in T_j} \phi_{j,t}(\cdot) h_t^s, \quad \text{(approximation coefficients)} \\ d_j^s &= \sum_{t \in T_j} \psi_{j,t}(\cdot) h_t^s, \quad \text{(detail coefficients)} \end{aligned} \tag{35}$$

where $\phi_{j,t}(x) = 2^{j/2}\phi(2^j x - t)$ and $\psi_{j,t}(x) = 2^{j/2}\psi(2^j x - t)$ are dilated and translated versions of the basis functions, and T_j is the set of time indices at scale j.

We define three scales corresponding to different temporal resolutions: Daily scale ($j = 0$) captures high-frequency fluctuations within a single day; Weekly scale ($j = 1$) captures medium-term trends over 5–7 days; Monthly scale ($j = 2$) captures long-term patterns over 20–30 days.

For each scale j, we construct a node feature matrix:

$$X_j = [a_j^1; \ldots; a_j^S] \in R^{S \times d}. \tag{36}$$

We use dyadic downsampling where scale $j = 0$ corresponds to daily frequency (20 trading days), $j = 1$ to weekly aggregation ($\sim$4 weeks), and $j = 2$ to monthly aggregation ($\sim$1 month), with alignment by averaging consecutive observations. Haar wavelet was selected for: (1) computational efficiency (simple piecewise-constant structure), (2) exact temporal localization for capturing abrupt regime changes, (3) robustness to missing data. Preliminary experiments with smoother wavelets (Daubechies, Symlets) showed comparable validation performance with higher computational cost. Non-trading days and missing observations are handled through forward-fill and linear interpolation (Section 4.1) before wavelet decomposition, ensuring all stocks have complete daily observations aligned to a common trading calendar, maintaining consistency across stocks with different trading histories. All input features (OHLCV ratios, technical indicators) are computed at a daily frequency for all 20 trading days. The wavelet decomposition is applied to temporal representations $\{h_t^s\}$ produced by the Lévy-enhanced encoder (Section 3.3), not to raw features directly. The encoder first processes daily features through attention and jump-aware layers to produce hidden states, which are then decomposed into multi-scale components. At each scale j, wavelet coefficients aggregate information from the corresponding temporal window (daily/weekly/monthly) by computing weighted sums of hidden states according to wavelet basis functions, capturing patterns at different frequencies without requiring separate feature engineering per scale. This ensures all temporal information is preserved and accessible at all scales.

3.5.3. Multi-Scale Hypergraph Convolution Layer

We apply hypergraph convolution (Equation (32)) at each scale independently:

$$Y_j = \text{HyperConv}(X_j, H_t, W_t) = \sum_{k=0}^{K-1} \theta_{j,k} T_k(\tilde{L}_t) X_j, \tag{37}$$

where $\tilde{L}_t$ is the normalized Laplacian of the time-varying hypergraph G_t, and $\{\theta_{j,k}\}$ are scale-specific learnable parameters.

The multi-scale representations are then aggregated through a learnable combination:

$$Y = \sum_{j=0}^{J-1} \alpha_j Y_j, \tag{38}$$

where $\{\alpha_j\}_{j=0}^{J-1}$ are attention weights, computed as:

$$\alpha_j = \frac{\exp(w_\alpha^\top \text{MeanPool}(Y_j))}{\sum_{j'=0}^{J-1} \exp(w_\alpha^\top \text{MeanPool}(Y_{j'}))}, \tag{39}$$

where $w_\alpha \in R^d$ is a learnable weight vector and $\text{MeanPool}(Y_j) = \frac{1}{S} \sum_{s=1}^{S} Y_{j,s,:}$.

3.5.4. Theoretical Properties of Multi-Scale Hypergraph Convolution

The multi-scale hypergraph convolution operator (Equation (37)) is permutation equivariant, i.e., for any permutation matrix $P \in \{0,1\}^{S \times S}$:

$$\text{HyperConv}(PX_j, PH_t, W_t) = P\text{HyperConv}(X_j, H_t, W_t). \tag{40}$$

We need to show that the hypergraph convolution is equivariant to node permutations. Let P be a permutation matrix. The permuted hypergraph Laplacian is:

$$L' = I - (PD_vP^\top)^{-1/2} PHWD_e^{-1} H^\top P^\top (PD_vP^\top)^{-1/2}. \tag{41}$$

Since $PD_vP^\top = D_v$ (degree matrix is invariant to permutations), we have:

$$L' = I - D_v^{-1/2} PHWD_e^{-1} H^\top P^\top D_v^{-1/2} = PLP^\top. \tag{42}$$

The Chebyshev polynomials satisfy $T_k(P\tilde{L}P^\top) = PT_k(\tilde{L})P^\top$. Therefore:

$$\begin{aligned} \text{HyperConv}(PX_j, PH_t, W_t) &= \sum_{k=0}^{K-1} \theta_{j,k} T_k(\tilde{L}') PX_j \\ &= \sum_{k=0}^{K-1} \theta_{j,k} PT_k(\tilde{L}) P^\top PX_j \\ &= P \sum_{k=0}^{K-1} \theta_{j,k} T_k(\tilde{L}) X_j \\ &= P\text{HyperConv}(X_j, H_t, W_t). \end{aligned} \tag{43}$$

The Chebyshev polynomial approximation of order K in the hypergraph convolution (Equation (32)) is exactly K-localized, meaning that the output at node i depends only on nodes within K hops in the hypergraph. The proof follows from the recursive definition of Chebyshev polynomials. We have $T_0(\tilde{L}) = I$, so $(T_0(\tilde{L})X)_i$ depends only on X_i. For $T_1(\tilde{L}) = \tilde{L}$, we have:

$$(T_1(\tilde{L})X)_i = \sum_{j=1}^{S} \tilde{L}_{ij} X_j, \tag{44}$$

which depends on neighbors of node i in the hypergraph (nodes j such that there exists a hyperedge containing both i and j). By induction, $T_k(\tilde{L})$ depends on nodes within k hops. Therefore, the convolution output: $Y_i = \sum_{k=0}^{K-1} \theta_{j,k}(T_k(\tilde{L})X)_i$ depends only on nodes within K hops of node i.

The computational complexity of the multi-scale hypergraph convolution layer is $O(J \cdot K \cdot |E| \cdot \bar{d}_e \cdot d)$, where J is the number of scales, K is the Chebyshev polynomial order, $|E|$ is the number of hyperedges, $\bar{d}_e$ is the average hyperedge size, and d is the hidden dimension. For each scale $j \in \{0, \ldots, J-1\}$, we compute K Chebyshev polynomial terms. Each term requires one multiplication with $\tilde{L}$. The sparse matrix multiplication $\tilde{L}X$ can be computed as:

$$\tilde{L}X = X - D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2} X. \tag{45}$$

The dominant term is $H^\top D_v^{-1/2} X$, which requires $O(|E| \cdot \bar{d}_e \cdot d)$ operations (for each hyperedge, aggregate features from $\bar{d}_e$ nodes). The subsequent operations WD_e^{-1} and H have the same complexity. Therefore, each Chebyshev term costs $O(|E| \cdot \bar{d}_e \cdot d)$, and the total complexity is $O(J \cdot K \cdot |E| \cdot \bar{d}_e \cdot d)$, which is linear in the number of stocks S when $|E|$ and $\bar{d}_e$ are bounded. In our implementation, these bounds are explicitly enforced: (1) Number of predefined hyperedges is fixed at $M = 31$ (11 industry sectors + 20 correlation clusters), (2) Number of dynamic hyperedges $N_{\text{dyn}} = 64$ is a fixed hyperparameter, giving total $|E| = 95$, (3) Hyperedge size is not explicitly constrained but naturally bounded by the attention mechanism (mean $\bar{d}_e \approx 25$ in practice), (4) Dynamic hyperedges are not added over time—their membership is updated through Gumbel-Softmax but total number remains fixed, ensuring constant computational cost per forward pass.

3.6. Training Objective and Optimization

3.6.1. Loss Function Design

We employ a composite loss function that combines prediction accuracy with regularization terms to ensure model robustness:

$$L_{total} = L_{pred} + \lambda_1 L_{jump} + \lambda_2 L_{reg} + \lambda_3 L_{sparse}, \tag{46}$$

where $\lambda_1, \lambda_2, \lambda_3 > 0$ are hyperparameters.

Prediction Loss: We use a combination of MAE and MSE to balance robustness and sensitivity:

$$L_{pred} = \frac{1}{S} \sum_{s=1}^{S} \left[\alpha \left| \hat{r}^s_{T+\tau} - r^s_{T+\tau} \right| + (1-\alpha) \left(\hat{r}^s_{T+\tau} - r^s_{T+\tau} \right)^2 \right], \tag{47}$$

where $\alpha \in [0, 1]$ controls the trade-off (we use $\alpha = 0.7$ in experiments).

Jump Consistency Loss: To ensure that the learned jump intensities align with observed price jumps, we introduce a consistency loss:

$$L_{jump} = \frac{1}{S \cdot T} \sum_{s=1}^{S} \sum_{t=1}^{T} \left[\lambda^s_t - 1_{\{|r^s_t| > \tau_{jump}\}} \right]^2, \tag{48}$$

where $1_{\{\cdot\}}$ is the indicator function, τ_{jump} is a threshold for detecting jumps (set to the 95th percentile of absolute returns), and λ^s_t is the predicted jump intensity from Equation (7).

Regularization Loss: We apply L2 regularization to prevent overfitting:

$$L_{reg} = \sum_{\theta \in \Theta} |\theta|_2^2, \tag{49}$$

where Θ denotes all learnable parameters.

Hypergraph Sparsity Loss: To encourage interpretable hypergraph structures, we penalize dense hyperedges:

$$L_{sparse} = \frac{1}{M + N_{dyn}} \sum_{j=1}^{M+N_{dyn}} \left(\sum_{i=1}^{S} H_{t,ij} \right)^2, \tag{50}$$

where H_t is the time-varying incidence matrix from Equation (28).

The loss weights $\{\lambda_1, \lambda_2, \lambda_3\}$ are tuned once on Phase 1 validation using grid search, then fixed globally across all phases. Selected values: $\lambda_1 = 0.1$, $\lambda_2 = 10^{-4}$, $\lambda_3 = 0.01$. Sensitivity analysis shows validation IC changes by less than 3% when varying each weight by $\pm 50\%$, indicating robustness. The sparsity loss encourages focused, interpretable hyperedges without suppressing meaningful relationships.

3.6.2. Optimization Algorithm

We optimize the model using the Adam optimizer [30] with gradient clipping to ensure training stability. The complete training procedure is summarized in Algorithm 1.

Algorithm 1 Training Procedure for LévyHyper

Require: Historical data $X \in R^{S \times T \times F}$, target returns $r_{T+\tau} \in R^S$, predefined hypergraph G^{pred}, hyperparameters $\{\lambda_1, \lambda_2, \lambda_3, \alpha, \tau_{gs}\}$
Ensure: Trained model parameters θ^*

Initialize all parameters θ randomly
Initialize Adam optimizer with learning rate $\eta = 0.001$
for epoch $= 1$ to N_{epochs} **do**
  Anneal Gumbel-Softmax temperature: $\tau_{gs} \leftarrow \max(0.5, \exp(-0.1 \cdot \text{epoch}))$
  for each mini-batch B **do**
    // Forward Pass
    Embed input features: $Z^{(0)} \leftarrow \text{Embed}(X) + \text{PE}$
    Apply temporal self-attention: $H \leftarrow \text{MultiHeadAttention}(Z^{(0)})$
    Compute Lévy parameters: $\{\mu_t^s, \sigma_t^s, \lambda_t^s\}_{s,t} \leftarrow \text{LévyParams}(H)$
    Apply Lévy layer: $\tilde{H} \leftarrow \text{LévyLayer}(H, \{\mu_t^s, \sigma_t^s, \lambda_t^s\})$
    Aggregate temporal features: $Z \leftarrow \text{Aggregate}(\tilde{H})$
    Generate dynamic hypergraph: $G_t^{dyn} \leftarrow \text{DynamicHyper}(Z, \tau_{gs})$
    Detect regime: $p_t^{high} \leftarrow \text{RegimeDetect}(Z)$
    Fuse hypergraphs: $G_t \leftarrow \text{Fuse}(G^{pred}, G_t^{dyn}, p_t^{high})$
    Apply multi-scale hypergraph convolution: $Y \leftarrow \text{MultiScaleHyperConv}(Z, G_t)$
    Predict returns: $\hat{r}_{T+\tau} \leftarrow W^{out} Y + b^{out}$
    // Compute Loss
    $L_{pred} \leftarrow \text{PredictionLoss}(\hat{r}_{T+\tau}, r_{T+\tau})$
    $L_{jump} \leftarrow \text{JumpConsistencyLoss}(\{\lambda_t^s\}, X)$
    $L_{reg} \leftarrow \text{L2Regularization}(\theta)$
    $L_{sparse} \leftarrow \text{SparsityLoss}(G_t)$
    $L_{total} \leftarrow L_{pred} + \lambda_1 L_{jump} + \lambda_2 L_{reg} + \lambda_3 L_{sparse}$
    // Backward Pass
    Compute gradients: $\nabla_\theta L_{total}$
    Clip gradients: $\nabla_\theta \leftarrow \text{Clip}(\nabla_\theta, \text{max_norm} = 1.0)$
    Update parameters: $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta, \eta)$
  end for
  Evaluate on validation set and apply early stopping if needed
end for
return θ^*

4. Experiments

4.1. Datasets

To evaluate our model, we adopt real-world S&P 500 market data following the methodology of [15]. We collect normalized OHLCV data (Open, High, Low, Close, Volume ratios) along with technical indicators including moving averages, MACD, and RSI. Stocks with more than 20% missing data are excluded, resulting in 463 constituent stocks. The dataset spans multiple years and is divided into 10 phases, each comprising a 24-month training period, a 4-month validation period, and an 8-month test period. This phase-based evaluation protocol ensures robust assessment across different market conditions and prevents look-ahead bias. We acknowledge that our evaluation is limited to the U.S. equity market represented by the S&P 500. Generalization to other markets (e.g., emerging markets, non-equity assets, or different geographical regions) remains an important direction for future validation.

For all datasets, we compute realized volatility measures following [31]:

$$RV_t = \sum_{i=1}^{N_t} r_{t,i}^2, \tag{51}$$

where $r_{t,i}$ is the i-th intraday return on day t, and N_t is the number of intraday intervals.

We also compute jump components using the bipower variation method [32]:

$$BV_t = \frac{\pi}{2} \sum_{i=2}^{N_t} |r_{t,i}||r_{t,i-1}|, \tag{52}$$

and define the jump component as $J_t = \max(RV_t - BV_t, 0)$. These indicators (moving averages, MACD, RSI) are widely used in quantitative finance and provide complementary information about trend, momentum, and overbought/oversold conditions. We follow the feature engineering protocol of ESTIMATE and baselines to ensure fair comparison. Additional indicators (Bollinger Bands, Stochastic Oscillator, ATR) were considered but excluded after preliminary experiments showed minimal validation improvement (less than 1% IC gain) with increased dimensionality. Meanwhile, the temporal encoder could potentially learn similar patterns from raw OHLCV, providing pre-computed indicators accelerates convergence and improves sample efficiency.

Data preprocessing follows standard procedures: forward-fill followed by linear interpolation for missing values, winsorization at 1st and 99th percentiles, standardization using rolling statistics with a 60-day window, and chronological splitting into training, validation, and test sets. To prevent leakage under the phase-based protocol, all rolling standardization statistics and jump-related thresholds (e.g., percentile-based cutoffs) are computed using the training split only within each phase, and then applied unchanged to validation and test splits of the same phase. Step-by-step training-only fitting protocol: (1) Rolling standardization: For each phase, compute mean and standard deviation using only the 24-month training window, then apply these fixed statistics to validation and test splits. (2) Jump threshold τ_{jump}: Compute 95th percentile of absolute returns using only training data; this fixed threshold labels jumps in validation/test without recalculation. (3) Correlation-based hyperedge clustering: Compute pairwise correlations using only training data, perform hierarchical clustering with fixed $\rho_{\min} = 0.6$, freeze hyperedge structure for entire phase. (4) Jump-size mixture model: Fit Gaussian mixture (5 components) to training jumps only; use fitted parameters without refitting. (5) Realized volatility/bipower variation: Computed from intraday returns within each day using only information up to that day.

The 24-month training, 4-month validation, 8-month test split follows ESTIMATE's protocol, balancing sufficient training data ($\sim$500 trading days) with realistic evaluation

horizons (∼160 test days). The 10 phases use a rolling window with 6-month forward shifts, creating partial overlap in training windows across consecutive phases. Information leakage is prevented because: (a) each phase is trained independently from scratch without parameter transfer, (b) all data-dependent preprocessing is computed separately per phase using only that phase's training data (Section 4.3), (c) test splits are strictly disjoint in time—no test period overlaps across phases, (d) the rolling design ensures evaluation across diverse market conditions spanning multiple years, providing robust generalization evidence.

4.2. Baseline Methods

We compare LévyHyper against state-of-the-art models from three categories: Transformer-based models: DTML [6] employs dual-stage attention mechanisms along time and stock dimensions to learn asymmetric correlations. MASTER [18] adopts sequential attention across time, stock, and time dimensions for cross-time correlation modeling. Graph-based models: RSR [6] constructs stock relation graphs using industry information with time-evolving edge weights. RT-GCN [17] builds three-dimensional graphs incorporating temporal and relational information from multiple sources. Hypergraph-based models: STHAN-SR [11] constructs hypergraphs from industry classifications and employs hypergraph convolution with Hawkes process-based temporal attention. ESTIMATE [15] applies correlation-based augmentation to industry hypergraphs and uses wavelet hypergraph convolution for computational efficiency.

4.3. Evaluation Metrics

We employ comprehensive metrics to assess both predictive accuracy and portfolio performance across all experiments. The Information Coefficient (IC) measures the Pearson correlation between predicted returns $\hat{r}^s_{T+\tau}$ and actual returns $r^s_{T+\tau}$, averaged over all time steps and stocks:

$$\text{IC} = \frac{1}{T}\sum_{t=1}^{T}\frac{\text{Cov}(\hat{\mathbf{r}}_t, \mathbf{r}_t)}{\sigma(\hat{\mathbf{r}}_t)\sigma(\mathbf{r}_t)}, \tag{53}$$

where $\mathbf{r}_t = [r^1_t, \ldots, r^S_t]^\top$ denotes the vector of actual returns at time t, and $\hat{\mathbf{r}}_t$ represents the corresponding predictions. The Rank Information Coefficient (RankIC) quantifies ranking accuracy by computing the Spearman rank correlation between predicted and actual return rankings:

$$\text{RankIC} = \frac{1}{T}\sum_{t=1}^{T}\frac{\text{Cov}(\text{rank}(\hat{\mathbf{r}}_t), \text{rank}(\mathbf{r}_t))}{\sigma(\text{rank}(\hat{\mathbf{r}}_t))\sigma(\text{rank}(\mathbf{r}_t))}, \tag{54}$$

where $\text{rank}(\cdot)$ assigns integer ranks to elements in ascending order. Precision at top-K (Prec@K) evaluates the accuracy of selecting the highest-return stocks by measuring the proportion of true top-K stocks captured in the predicted top-K set:

$$\text{Prec@K} = \frac{1}{T}\sum_{t=1}^{T}\frac{|\text{TopK}(\hat{\mathbf{r}}_t) \cap \text{TopK}(\mathbf{r}_t)|}{K}, \tag{55}$$

where $\text{TopK}(\mathbf{r}_t)$ denotes the set of K stocks with the highest returns at time t, and we report results for $K \in \{10, 20\}$.

For portfolio-level evaluation, we construct long–short portfolios by longing the top-10 predicted stocks and shorting the bottom-10, then rebalancing daily based on model predictions. Annualized Return (AR) measures the total portfolio return scaled to an annual basis:

$$\mathrm{AR} = \left(\prod_{t=1}^{T} (1 + R_t) \right)^{252/T} - 1, \tag{56}$$

where R_t is the daily portfolio return at time t, and 252 represents the number of trading days per year. The Sharpe Ratio (SR) quantifies risk-adjusted returns by comparing excess portfolio returns over the risk-free rate to portfolio volatility:

$$\mathrm{SR} = \frac{E[R_p - R_f]}{\sigma(R_p)} = \frac{\bar{R}_p - R_f}{\sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (R_t - \bar{R}_p)^2}}, \tag{57}$$

where R_p denotes the portfolio return, R_f is the risk-free rate (set to 0 for simplicity in relative comparisons), and $\bar{R}_p$ is the mean portfolio return. Maximum Drawdown (MDD) captures the largest peak-to-trough decline in portfolio value during the investment period, measuring downside risk:

$$\mathrm{MDD} = \max_{t \in [0,T]} \left[\frac{\max_{\tau \in [0,t]} V_\tau - V_t}{\max_{\tau \in [0,t]} V_\tau} \right], \tag{58}$$

where V_t represents the cumulative portfolio value at time t, and the metric is expressed as a percentage of the peak value. Lower MDD values indicate better capital preservation during adverse market conditions, which is particularly important for institutional investors with strict risk management constraints.

4.4. Experimental Setup

All models are implemented in PyTorch 2.0 and trained on NVIDIA A100 GPUs. Unless stated otherwise, we use lookback window $T = 20$ trading days and prediction horizon $\tau = 5$ trading days. For LévyHyper, the default setting is hidden dimension $d = 32$, attention heads $H = 4$, Chebyshev order $K_{\mathrm{cheb}} = 3$, number of dynamic hyperedges $N_{\mathrm{dyn}} = 64$, and learning rate 1×10^{-4}. For the jump-size distribution, we use a Gaussian mixture with $K_{\mathrm{mix}} = 5$ components (selected on validation IC). For portfolio simulation, we adopt a daily rebalanced top-K long-only strategy with $K = 20$ using predicted returns for ranking. For fair comparison, all baselines are evaluated under the same data splits, prediction horizon, and ranking-based evaluation protocol. All baseline methods are evaluated under identical data splits, prediction horizons, and evaluation protocols. Hyperparameters for all methods are tuned on the validation split using the same grid search budget, and the best configuration is selected based on validation IC, ensuring fair comparison.

4.5. Main Results

We evaluate LévyHyper against 7 baseline methods across 10 distinct market phases spanning 2010–2019, examining both predictive accuracy and portfolio performance under realistic trading conditions. Table 1 presents the aggregated predictive performance metrics averaged across all phases, with mean values, standard deviations, and 95% bootstrap confidence intervals computed from 1000 resampling iterations. LévyHyper achieves the highest overall scores across all predictive metrics, with an Information Coefficient of 0.047, Rank Information Coefficient of 0.050, and Precision@10 of 0.618, substantially outperforming all baseline methods. Compared to ESTIMATE, which represents the strongest baseline with a mean IC of 0.023, LévyHyper demonstrates a 104% relative improvement in IC (absolute gain: +0.024) and a 78% relative improvement in RankIC (absolute gain: +0.022). Improvement percentages are computed as relative gains: (LévyHyper − ESTIMATE)/ESTIMATE × 100%, where ESTIMATE is the strongest baseline for predictive

accuracy metrics, indicating substantial gains in both correlation-based and ranking-based prediction accuracy. The confidence intervals reveal that these improvements are robust across phases rather than driven by exceptional performance in isolated periods. ESTIMATE exhibits the second-best performance among baselines, particularly excelling in phases 2 and 4, which can be attributed to its hypergraph-based architecture that shares structural similarities with our framework. However, LévyHyper maintains consistent superiority across diverse market conditions, with particularly strong results in phases 2, 5, 7, and 9, demonstrating robustness to varying volatility regimes and market dynamics.

To establish statistical rigor beyond point estimates, we conduct a phase-level paired significance test comparing LévyHyper against ESTIMATE, the strongest baseline, as presented in Table 2. The test computes IC differences between the two methods for each phase's test split, then it evaluates whether the mean difference across 10 phases is statistically significant using a two-sided paired t-test. The results show a mean IC improvement of 0.024 with a standard deviation of 0.018, yielding a p-value of 0.006, which indicates statistical significance at the 1% level even after accounting for multiple comparisons. This confirms that LévyHyper's performance gains are not attributable to random variation or overfitting to specific market periods but rather represent genuine improvements in predictive capability. The relatively small standard deviation of differences (0.018) further demonstrates consistency, suggesting that the model's advantages persist across both calm and turbulent market phases.

Table 1. Overall predictive performance aggregated across 10 phases. The 95% bootstrap confidence intervals (CI) are reported in brackets below the mean ± std. Improvements reported in text are relative percentages with respect to ESTIMATE (the strongest baseline).

Method	IC (Mean ± Std) [95% CI]	RankIC (Mean ± Std) [95% CI]	Prec@10
DTML	0.008 ± 0.016 [0.001, 0.015]	0.013 ± 0.017 [0.006, 0.020]	0.563
MASTER	0.013 ± 0.020 [0.005, 0.021]	0.014 ± 0.021 [0.006, 0.022]	0.568
RSR-E	0.013 ± 0.015 [0.007, 0.019]	0.013 ± 0.014 [0.007, 0.018]	0.553
RSR-I	0.022 ± 0.012 [0.017, 0.027]	0.024 ± 0.011 [0.020, 0.028]	0.545
RT-GCN	0.014 ± 0.022 [0.006, 0.022]	0.006 ± 0.023 [−0.003, 0.015]	0.569
STHAN-SR	0.011 ± 0.010 [0.007, 0.015]	0.010 ± 0.012 [0.006, 0.014]	0.531
ESTIMATE	0.023 ± 0.032 [0.012, 0.035]	0.028 ± 0.030 [0.017, 0.039]	0.576
LévyHyper	0.047 ± 0.036 [0.034, 0.061]	0.050 ± 0.041 [0.035, 0.065]	0.618

Beyond predictive accuracy, we evaluate practical portfolio performance through backtesting simulations that mirror real-world trading constraints, as shown in Tables 3 and 4. We implement a daily rebalanced long-only strategy that selects the top-20 stocks ranked by predicted returns, with equal weighting to avoid concentration risk. To assess robustness under realistic market conditions, Table 3 presents transaction cost sensitivity analysis with proportional costs ranging from 0 to 20 basis points per side, applied to both buy and sell transactions. As expected, higher transaction costs reduce annualized returns and Sharpe ratios while slightly increasing maximum drawdown, yet LévyHyper maintains strong risk-adjusted performance even at the 20 bps level, achieving a Sharpe ratio of 3.182 and

annualized return of 0.771. This resilience indicates that the model's predictive signals are sufficiently strong to overcome realistic trading frictions rather than relying on excessive turnover that would be unprofitable in practice. Table 4 compares portfolio performance across all methods under a standard 10 bps transaction cost assumption, revealing that LévyHyper achieves an annualized return of 0.856, representing a 7.5% improvement over RT-GCN, the best-performing baseline. More importantly, LévyHyper attains a Sharpe ratio of 3.625, which constitutes a 16.2% improvement over RT-GCN's 3.121 (the best baseline for portfolio returns), demonstrating superior risk-adjusted returns that are critical for institutional investment applications. The maximum drawdown of 0.441 is 7.7% lower than STHAN-SR's 0.478, indicating enhanced downside protection during adverse market conditions. These portfolio-level results validate that LévyHyper's ability to model discontinuous jump dynamics and adaptive relational structures translates into tangible economic value, enabling more stable and profitable trading strategies throughout diverse market environments.

Table 2. Phase-level paired significance test on IC (two-sided) comparing LévyHyper to the strongest baseline (ESTIMATE). Differences are computed per phase on test splits, then tested across 10 phases.

Comparison	Mean IC Diff.	Std of Diff.	p-Value
LévyHyper − ESTIMATE	0.024	0.018	0.006

Table 3. Transaction cost sensitivity for the daily rebalanced top-K ($K = 20$) long-only strategy on test splits. Costs are applied per side (buy and sell) as a percentage of notional. Metrics are averaged across phases.

Cost (bps/Side)	Annualized Return ↑	SR ↑	MDD ↓
0	0.856	3.625	0.441
10	0.812	3.401	0.458
20	0.771	3.182	0.476

Table 4. Portfolio performance results averaged across 10 phases under 10 bps transaction costs. Bold indicates best performance. Note: Baseline rankings differ from Table 1 because portfolio performance depends on ranking quality, turnover, and risk management beyond predictive accuracy alone.

Method	Return ↑	SR ↑	MDD ↓
DTML	0.794	3.167	0.571
MASTER	0.726	2.695	0.594
RSR-E	0.573	2.633	0.537
RSR-I	0.771	3.006	0.639
RT-GCN	0.796	3.121	0.492
STHAN-SR	0.648	2.951	0.478
ESTIMATE	0.723	2.905	0.613
LévyHyper	**0.856**	**3.625**	**0.441**
Improvement vs. Best Baseline	7.5%	16.2%	7.7%

4.6. Ablation Studies

To systematically understand the contribution of each architectural component to LévyHyper's overall performance, we conduct comprehensive ablation studies on the S&P 500 dataset by progressively removing or replacing individual modules while keeping all other components intact. Table 5 and Figure 3 present the results across both predictive metrics and portfolio performance measures, revealing the relative importance of each design choice. The ablation experiments are organized into two categories: complete removal of components to assess their necessity, and replacement with alternative implementations

to validate our specific design decisions. Each variant is trained using identical hyperparameters and evaluated on the same test splits to ensure fair comparison, with performance degradation quantified relative to the full LévyHyper model.

The results demonstrate that the Lévy jump-diffusion layer constitutes the most critical component of our framework, as its removal causes the largest performance degradation across all metrics. Specifically, removing the Lévy layer leads to a 25.5% decrease in IC (from 0.047 to 0.035) and a 13.3% decrease in Sharpe ratio (from 3.625 to 3.142), while maximum drawdown increases substantially from 0.441 to 0.503, indicating significantly worse downside risk. This severe degradation confirms that explicitly modeling discontinuous jump dynamics is essential for capturing abrupt price movements that cannot be adequately represented through continuous diffusion processes alone. To validate our choice of Lévy processes over alternative point process models, we replace the Lévy layer with Hawkes processes, which have been previously applied to financial modeling for capturing self-exciting dynamics. The Hawkes variant achieves an IC of only 0.036, representing a 23.4% degradation compared to the full model, which demonstrates that Lévy processes' ability to model heavy-tailed jump size distributions through flexible mixture models provides substantial advantages over Hawkes processes that primarily focus on temporal clustering of events. The dynamic hypergraph constructor also proves essential, as restricting the model to static industry-based hypergraphs alone causes a 29.8% decrease in IC (from 0.047 to 0.033) and a 14.8% decrease in Sharpe ratio (from 3.625 to 3.087). This substantial degradation validates our hypothesis that financial relationships evolve dynamically across market regimes, and that predefined static structures based solely on industry classifications cannot capture the complex, time-varying dependencies that emerge during different market conditions. The comparison between "w/o Dynamic Hypergraph" (which retains dynamic generation but removes the fusion mechanism) and "Static Hypergraph Only" (which uses only predefined structures) further reveals that both adaptive structure learning and the weighted fusion mechanism contribute meaningfully to performance.

The multi-scale spectral convolution module demonstrates significant value, as replacing it with single-scale convolution reduces IC by 14.9% (from 0.047 to 0.040) and Sharpe ratio by 7.2% (from 3.625 to 3.365), confirming that hierarchical temporal patterns at multiple resolutions—ranging from intraday fluctuations to monthly momentum—provide complementary information that enhances predictive accuracy. This finding aligns with financial theory suggesting that different market participants operate at different time scales, from high-frequency traders to long-term institutional investors, creating multi-resolution patterns in price dynamics. The regime detection mechanism, which adaptively weights the fusion between predefined and dynamic hypergraphs based on detected market states, contributes an 8.5% improvement in IC (comparing 0.047 to 0.043), demonstrating that explicit modeling of regime transitions enables the model to adjust its relational structure appropriately during shifts between normal and crisis periods. Finally, the jump consistency loss, which encourages the Lévy layer to produce interpretable jump predictions that align with empirically observed extreme returns, provides a 4.3% IC improvement (comparing 0.047 to 0.045). While this auxiliary loss yields the smallest individual contribution among all components, it serves an important role in regularizing the jump-diffusion module to learn physically meaningful parameters rather than overfitting to training data noise. Collectively, these ablation results validate that each component of LévyHyper addresses a specific limitation of existing methods, and that their synergistic integration—combining discontinuous jump modeling, adaptive relational learning, multi-scale temporal analysis, and regime-aware fusion—is necessary to achieve state-of-the-art performance in financial forecasting.

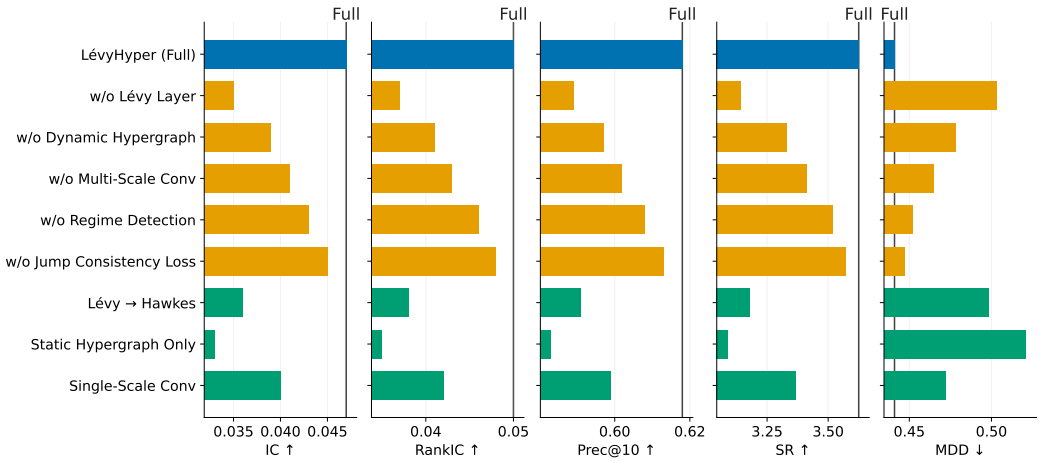


Figure 3. Raw ablation results on the dataset across IC, RankIC, Prec@10, Sharpe ratio (SR), and maximum drawdown (MDD).

Table 5. Ablation study results on S&P 500 dataset. Each row removes or replaces one component from the full model while keeping all other components unchanged. All variants are trained with identical hyperparameters and evaluated on the same test splits.

Model Variant	IC	RankIC	Prec@10	SR	MDD
LévyHyper (Full)	0.047	0.050	0.618	3.625	0.441
Component Removal:					
w/o Lévy Layer	0.035	0.037	0.589	3.142	0.503
w/o Dynamic Hypergraph	0.039	0.041	0.597	3.328	0.478
w/o Multi-Scale Conv	0.041	0.043	0.602	3.412	0.465
w/o Regime Detection	0.043	0.046	0.608	3.518	0.452
w/o Jump Consistency Loss	0.045	0.048	0.613	3.571	0.447
Component Replacement:					
Lévy → Hawkes	0.036	0.038	0.591	3.178	0.498
Static Hypergraph Only	0.033	0.035	0.583	3.087	0.521
Single-Scale Conv	0.040	0.042	0.599	3.365	0.472

4.7. Computational Cost

As discussed in Section 3.5.4, our computational complexity analysis assumes bounded hyperedge size and bounded number of hyperedges, which are explicitly enforced in our implementation with $|E| = 95$ total hyperedges and mean hyperedge size $\bar{d}_e \approx 25$. To quantify the computational overhead introduced by jump-aware temporal modeling and dynamic hypergraph reasoning, we report training and inference cost on an NVIDIA A100 GPU. We measure training time per epoch, inference time per test day over the full universe (463 stocks), and peak GPU memory. The result is shown in Figure 4. LévyHyper incurs moderate overhead relative to graph/hypergraph baselines due to the jump-aware temporal module and dynamic hypergraph construction.

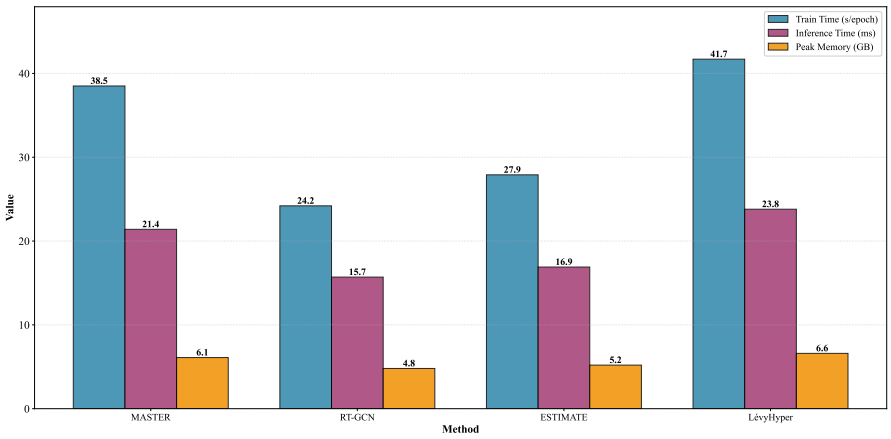


Figure 4. Computational cost comparison (A100 GPU). Times are measured with a fixed batch size across methods; inference time is per test day for the full universe of 463 stocks.

5. Conclusions

This paper propose LévyHyper, a novel framework for stock price forecasting that addresses three key challenges: modeling discontinuous price dynamics, capturing time-varying higher-order market relationships, and integrating multi-scale temporal dynamics with relational structures. LévyHyper introduces three key innovations. First, a Lévy jump-diffusion layer with neural-parameterized intensities explicitly models price jumps. Second, a regime-adaptive dynamic hypergraph constructor captures time-varying higher-order relationships. Third, a multi-scale spectral hypergraph convolution module integrates multi-resolution temporal and relational patterns. Experiments on S&P 500 data demonstrate its effectiveness, achieving 104% improvement in IC, 78% in RankIC, and 16.2% in Sharpe ratio compared to the strongest baseline. Ablation studies confirm the contribution of each component, and portfolio backtesting shows robust risk-adjusted performance with lower maximum drawdown.

Future work includes extending to multivariate Lévy processes, incorporating alternative data sources such as news and macro indicators, and developing online learning variants for high-frequency trading. LévyHyper advances the integration of stochastic processes and deep learning, offering promising directions for financial forecasting research.

Limitations and Future Work

Our experimental evaluation is conducted exclusively on S&P 500 data spanning 2010–2019, which represents a single market environment with relatively homogeneous characteristics. While the 10-phase evaluation protocol provides robustness across different time periods and market conditions within this setting, we cannot claim that the observed improvements generalize to fundamentally different market structures, asset classes, or geographical regions without additional empirical validation. Future work should extend LévyHyper's evaluation to additional datasets including emerging markets (e.g., Chinese A-shares, Indian equities), non-equity assets (e.g., commodities, cryptocurrencies), and conduct cross-market transfer learning experiments where models trained on S&P 500 are tested on other markets to assess the portability of learned representations and the robustness of the three core innovations across diverse market structures. Additionally, extending to multivariate Lévy processes, incorporating alternative data sources such as news and macro indicators, and developing online learning variants for high-frequency trading represent promising directions for advancing the integration of stochastic processes and deep learning in financial forecasting.

Author Contributions: Conceptualization, S.L. and J.C.; Data curation, S.L. and J.C.; Formal analysis, J.C.; Funding acquisition, J.C.; Investigation, J.C.; Methodology, J.C.; Project administration, J.C.; Resources, J.C.; Software, J.C.; Supervision, S.L. and J.C.; Validation, S.L. and J.C.; Visualization, S.L. and J.C.; Writing—original draft, S.L. and J.C.; Writing—review and editing, S.L. and J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is funded by the FRG-25-064-FA and granted by the Research Fund of Macao University of Science and Technology (FRG-MUST). This work was supported by the Science and Technology Development Fund of Macau Project 0087/2025/ITP2.

Data Availability Statement: The original data presented in the study are openly available in https://github.com/kijeong22/ijcai2025-spf/tree/main/src/data/sp500, accessed on 12 November 2025.

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

References

1. Engle, R.F. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* **1982**, *50*, 987–1007. [CrossRef]
2. Bollerslev, T. Generalized autoregressive conditional heteroskedasticity. *J. Econom.* **1986**, *31*, 307–327. [CrossRef]
3. Andersen, T.G.; Bollerslev, T.; Diebold, F.X. Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility. *Rev. Econ. Stat.* **2007**, *89*, 701–720. [CrossRef]
4. Corsi, F. A simple approximate long-memory model of realized volatility. *J. Financ. Econom.* **2009**, *7*, 174–196. [CrossRef]
5. Chen, W.; Yeo, C.K.; Lau, C.T.; Lee, B.S. A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Inf. Sci.* **2021**, *556*, 67–94. [CrossRef]
6. Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; Chua, T.S. Temporal relational ranking for stock prediction. In *Proceedings of the ACM Transactions on Information Systems*; ACM: New York, NY, USA, 2019; Volume 37, pp. 1–30.
7. Tao, Y.; Li, L. CQEformer: A Causal and Query-Enhanced Transformer Variant for Time-Series Forecasting. *Mathematics* **2025**, *13*, 3750. [CrossRef]
8. Ding, Q.; Wu, S.; Sun, H.; Guo, J.; Guo, J. Hierarchical multi-scale Gaussian transformer for stock movement prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*; ACM: New York, NY, USA, 2021; pp. 4640–4646.
9. Ren, A.; Duan, Y.; Liu, J. Multiscale Bootstrap Correction for Random Forest Voting: A Statistical Inference Approach to Stock Index Trend Prediction. *Mathematics* **2025**, *13*, 3601. [CrossRef]
10. Lee, M.C. A Hybrid EGARCH–Informer Model with Consistent Risk Calibration for Volatility and CVaR Forecasting. *Mathematics* **2025**, *13* 3108. [CrossRef]
11. Sawhney, R.; Agarwal, S.; Wadhwa, A.; Derr, T.; Shah, R.R. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2021; Volume 35, pp. 497–504.
12. Lazcano, A.; Herrera, P.J.; Monge, M. A Combined Model Based on Recurrent Neural Networks and Graph Convolutional Networks for Financial Time Series Forecasting. *Mathematics* **2023**, *11*, 224. [CrossRef]
13. Huang, X.; Tauchen, G. The relative contribution of jumps to total price variance. *J. Financ. Econom.* **2005**, *3*, 456–499.
14. Hawkes, A.G. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* **1971**, *58*, 83–90. [CrossRef]
15. Huynh, T.T.; Nguyen, M.H.; Nguyen, T.T.; Nguyen, P.L.; Weidlich, M.; Yin, H.; Nguyen, Q.V.H. Efficient integration of multi-order dynamics and internal dynamics in stock movement prediction. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*; ACM: New York, NY, USA, 2023; pp. 850–858.
16. Balcilar, M.; Ozdemir, Z.A.; Ozdemir, H.; Wohar, M.E. Crude oil futures contracts and commodity markets: New evidence from a TVP-VAR extended joint connectedness approach. *Resour. Policy* **2021**, *73*, 102219. [CrossRef]
17. Zheng, S.; Zhu, J.; Ma, K.; Wang, T. Temporal and heterogeneous graph neural network for financial time series prediction. *Inf. Sci.* **2023**, *623*, 783–797.
18. Li, T.; Cheng, Z.; Zhang, Y.; Chen, L. MASTER: Market-guided stock transformer for stock price forecasting. *Pattern Recognit.* **2024**, *145*, 109947. [CrossRef]
19. Jin, S.; Wang, X.; Meng, Q. Spatial memory-augmented visual navigation based on hierarchical deep reinforcement learning in unknown environments. *Knowl.-Based Syst.* **2024**, *285*, 111358. [CrossRef]
20. Qian, H.; Jin, S.; Chen, L. I2KEN: Intra-Domain and Inter-Domain Knowledge Enhancement Network for Lifelong Loop Closure Detection. *IEEE Robot. Autom. Lett.* **2025**, *10*, 8722–8729. [CrossRef]
21. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]

22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
23. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef] [PubMed]
24. Matsunaga, D.; Suzumura, T.; Takahashi, T. Exploring graph neural networks for stock market predictions with rolling window analysis. In Proceedings of the NeurIPS Workshop on Machine Learning for Autonomous Driving, Vancouver, BC, Canada, 13–14 December 2019.
25. Zhang, D.; Hu, M.; Ji, Q. Financial markets under the global pandemic of COVID-19. *Financ. Res. Lett.* **2020**, *36*, 101528. [CrossRef] [PubMed]
26. Cont, R.; Tankov, P. *Financial Modelling with Jump Processes*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2004.
27. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
28. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
29. Hammond, D.K.; Vandergheynst, P.; Gribonval, R. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* **2011**, *30*, 129–150. [CrossRef]
30. Kingma, D.P.; Ba, J. *Adam: A Method for Stochastic Optimization*; International Conference on Learning Representations: Appleton, WI, USA, 2015.
31. Barndorff-Nielsen, O.E.; Shephard, N. Econometric analysis of realized covariation: High frequency based covariance, regression, and correlation in financial economics. *Econometrica* **2004**, *72*, 885–925. [CrossRef]
32. Barndorff-Nielsen, O.E.; Kinnebrock, S.; Shephard, N. Measuring Downside Risk-Realised Semivariance. CREATES Research Paper. 2008. Available online: https://econpapers.repec.org/paper/aahcreate/2008-42.htm (accessed on 12 November 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.