The CSI Journal on
Computer Science and Engineering
Vol. 19, No. 2, 2025
Pages 1-10
doi.org/10.22034/jcse.2025.225717
Regular Paper

Volume-Enhanced Stock Price Forecasting: Multivariate Approach with Large Language Models

Sajad Mirzababaei[1] Moslem Habibi[1]

[1]Sharif University of Technology

Abstract

Accurate forecasting of stock prices and trading volumes underpins informed investment decisions and robust risk management. We present GPTTS, a transformer-based framework that integrates a specialized UnifiedDiffTokenizer for return-based discretization, parallel T5-derived price and volume encoders with lightweight adaptation (LoRA and adapters), and a gated cross-attention fusion module for cross-modal reasoning. On a comprehensive 15-year NASDAQ dataset, GPTTS delivers improvements of approximately 3% in MAE and 25% in MAPE over the strongest classical and deep-learning baselines, while reducing trainable parameters. These gains demonstrate the value of combining financial-aware tokenization with efficient fine-tuning to achieve high-fidelity, resource-conscious forecasting for algorithmic trading and risk analytics.

Keywords: Stock Price Forecasting, Large Language Models (LLMs), Multivariate Time-Series, Machine Learning in Finance

1. Introduction

Accurate time-series forecasting is crucial in financial markets, where the ability to predict stock prices and trading volumes directly influences investment strategies [1], risk management, and economic policy decisions. Financial time series, however, present a unique set of challenges due to their inherent complexity, characterized by high volatility, non-linearity, and the influence of various interrelated factors, including economic indicators, investor sentiment, and global events. Traditional forecasting approaches often fall short in capturing these complexities, particularly when both price and volume data are considered. These variables, while interdependent, provide different perspectives on market dynamics that are essential for making well-informed decisions.

Historically, models such as Autoregressive Integrated Moving Average (ARIMA) [2] and Generalized Autoregressive Conditional Heteroskedasticity(GARCH) [3] have been foundational in time-series forecasting. However, these models are limited by their reliance on linear assumptions, making them inadequate for capturing the non-linear patterns that are characteristic of financial time series. Furthermore, they are primarily designed for univariate forecasting and do not account for the potential interactions between multiple variables, such as price and volume.

Deep learning methods [4], particularly Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks [5] and Gated Recurrent Units (GRUs) [6], offer significant improvements by modelling temporal dependencies in sequential data. Despite these advancements, these models often struggle with long-term dependencies and require extensive feature engineering when dealing with multivariate data, which limits their effectiveness in complex financial forecasting scenarios.

More recently, transformer-based [7] models such as Chronos [8], have emerged as powerful tools for time-series forecasting, leveraging the self-attention mechanism to capture long-range dependencies. While Chronos represents a significant advancement over previous models, it primarily focuses on univariate time-series forecasting and does not attend to the crucial aspect of volume, which is vital in understanding stock market dynamics. Additionally, Chronos is designed as a general-purpose time-series forecasting model, lacking specific optimization for financial data, particularly stock market forecasting.

Recognizing these limitations, our research extends the capabilities of the Chronos model by incorporating both price

and volume data into the forecasting process. We propose a novel method, GPTTS (Generative Pretrained TimeSeries), that enhances the transformer-based encoder-decoder framework through a dual-stream architecture and a gated cross-attention fusion mechanism. This design allows the model to capture complex interdependencies between financial signals and improves forecast reliability under volatile market conditions. To achieve this, we introduce a unified tokenization strategy based on percentage returns and scaled volume changes, aligning with financial theory and enabling compatibility with language modelling approaches.

Moreover, GPTTS leverages modern training techniques, including Low-Rank Adaptation (LoRA), adapter layers, and quantization-aware strategies, to ensure scalability and efficiency. We train the model on a comprehensive 15-year NASDAQ dataset comprising more than 23 million records from over 7,000 stocks. This robust dataset ensures exposure to diverse market behaviours, from high-volatility episodes to long-term structural shifts.

Experimental results evaluated using performance metrics such as MAE, RMSE, and MAPE, demonstrate that GPTTS significantly outperforms existing models, including classical baselines and Chronos. These improvements are particularly evident in scenarios where both price and volume data are crucial for accurate forecasting. The integration of cross-attention mechanisms in the fusion process enhances the model's ability to exploit the intricate relationships between these variables, resulting in more nuanced and reliable predictions.

The remainder of this paper is structured as follows: Section 2, Related Works, reviews the existing body of literature on time-series forecasting, highlighting traditional statistical models, deep learning methods, and recent transformer-based architectures, with a focus on applications in financial contexts. Section 3, Data and Methods, describes the dataset construction and preprocessing steps, including scaling, quantization, and tokenization of price and volume data, along with the architectural design of the proposed GPTTS model—particularly its cross-attention mechanism for integrating tokenized representations. Section 4, Methodology, outlines the tokenizer design, fusion-based model architecture, and training strategies involving LoRA, adapter layers, and quantization. Section 5, Results, presents the evaluation outcomes comparing GPTTS against classical and state-of-the-art baselines using MAE, RMSE, and MAPE. Finally, Section 6, Conclusion, summarizes the main contributions, discusses the implications for financial forecasting, and suggests potential directions for future research.

2. Related Works

The forecasting of financial time series has long attracted research spanning classical statistical methods, neural architectures, and, more recently, transformer-based approaches. Effective prediction demands models that can capture complex temporal dependencies, non-linear patterns, and the interplay of heterogeneous signals such as price and trading volume. In this section, we briefly survey three broad categories of related work—statistical and econometric models, deep learning approaches, and transformer-based methods—before summarizing the remaining gaps that motivate our GPTTS model.

Statistical and Econometric Models. Early time-series forecasting relied on methods such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (SES, Holt–Winters) [9], and Generalized Autoregressive Conditional Heteroskedasticity (GARCH). While these models provide strong interpretability and well-understood theoretical properties, they assume linearity and stationarity, limiting their capacity to capture non-linear market dynamics or cross-variable interactions. Extensions like hybrid ARIMA–GARCH architectures [10] and wavelet-transformed LSTM frameworks [11] sought to mitigate some limitations by stacking non-linear components, but still treat price and volume largely as separate signals.

Deep Learning Approaches. The introduction of Recurrent Neural Networks (RNNs) [12] and their gated variants—Long Short-Term Memory (LSTM) [13] and Gated Recurrent Unit (GRU) [14]—enabled end-to-end learning of temporal dependencies. These models often outperform classical techniques on univariate tasks but require extensive feature engineering and struggle with long-range dependencies. More recent work employs attention mechanisms within sequence-to-sequence frameworks (e.g., Seq2Seq with attention [15]) to alleviate some limitations. Nevertheless, standard RNN-based architectures typically process price and volume either in concatenated feature vectors or via separate models without explicit fusion mechanisms, leaving cross-modality dynamics underexploited.

Transformer-Based Time-Series Models. Transformers, first introduced by Vaswani et al. [16], have reshaped sequence modelling by leveraging self-attention to capture long-range dependencies efficiently. Adaptations to time-series forecasting include the Temporal Fusion Transformer (TFT) [17], Informer [18], and Autoformer [19], which introduce enhancements like hierarchical attention and decomposition blocks. Chronos (T5-style sequence-to-sequence) applies a unified discretization tokenizer to return-based sequences but focuses primarily on price data, without jointly modelling dynamic features such as trading volume.

Tokenization and Efficient Fine-Tuning. Discretization of continuous signals into tokens has been shown to improve stability and leverage language-model strengths. Chronos tokenizer bins percentage returns into symbolic tokens, preserving statistical properties of financial time series. Meanwhile, parameter-efficient fine-tuning techniques—such as Low-Rank Adaptation (LoRA) [20] and adapter modules [21]—have emerged in NLP to reduce trainable parameters. However, existing forecasting works rarely combine advanced tokenization with these efficient adaptation methods in a multivariate settings.

Summary and Gaps. Although transformer-based models excel at capturing long-term dependencies, most prior work treats multivariate inputs simplistically, either ignoring volume or fusing heterogeneous signals without dedicated mechanisms. Moreover, the joint application of financial-aware tokenization and parameter-efficient adaptation remains unexplored. In response to these gaps, our GPTTS model introduces a dual-stream encoder for price and volume, a gated cross-attention fusion mechanism, a specialized UnifiedDiffTokenizer, and parameter-efficient fine-tuning via LoRA and adapters—addressing the limitations identified above.

3. Data

3.1. Dataset Overview

This study utilizes an accurate dataset comprising historical daily stock prices and trading volume data from NASDAQ-listed companies. Data were collected programmatically using the Yahoo Finance Python package (yfinance v0.2.x), which automatically adjusts for corporate actions such as stock splits and dividends. The dataset spans from January 2010 to November 2024, covering nearly 15 years of trading activity. This extended time horizon captures diverse market regimes, including secular bull runs, flash crashes, high-volatility events, and structural shifts such as the COVID-19 pandemic. Each stock's time series includes open, high, low, close (OHLC) prices, and corresponding daily traded volumes. In total, the dataset consists of 7,271 distinct equities and over 23 million records. The average series length is approximately 3,205 trading days, ensuring ample temporal depth for robust model training and evaluation.

3.2. Statistical Characteristics

Price Distribution. Figure 1 shows the distribution of stocks by closing price categories. The majority (65%) of stocks fall within the $10–$50 range, indicating a large concentration of mid-cap equities. Low-priced stocks below $10 account for 15% of the sample, while only 7.6% of stocks trade above $500.

Volume Distribution. Figure 2 depicts the log-scaled distribution of average daily trading volumes. The median log volume is $\log10(V) = 4.94$, corresponding to approximately 87,000 shares. The 25th percentile lies at $\log10(V) = 4.27$, and the 10th percentile at $\log10(V) = 3.64$, reflecting the presence of illiquid stocks alongside highly traded ones. Aggregate Metrics. Key aggregate statistics of the dataset are summarized below:

- Mean closing price: $866.18
- Daily return mean: 0.0202%

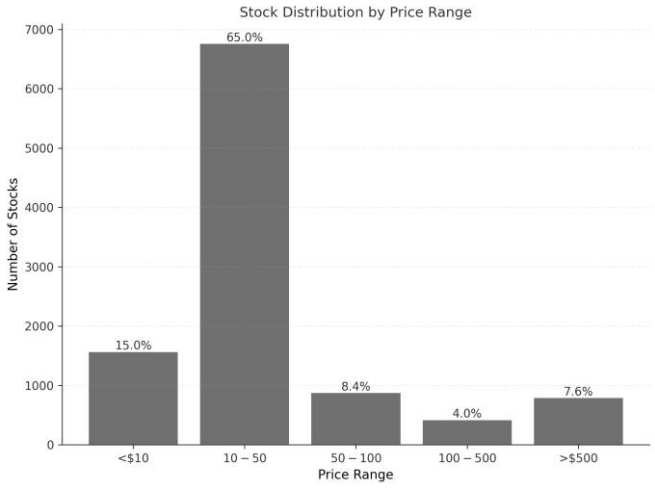

Figure 1: Distribution of stocks across closing price ranges. Mid-priced equities dominate the dataset.

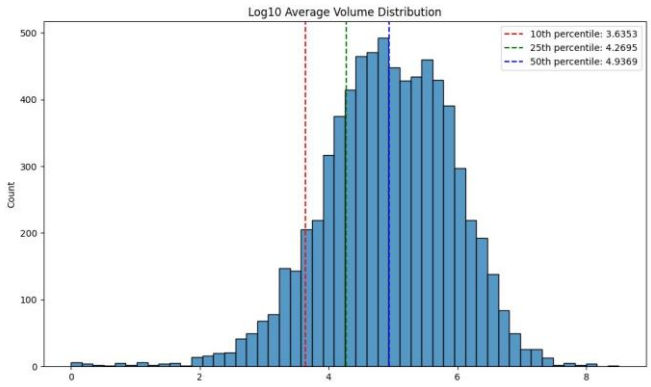

Figure 2: Histogram of log10-transformed average daily trading volumes. Dashed lines mark the 10th, 25th, and 50th percentiles.

- Standard deviation of returns: 2.96%
- 22-day rolling volatility (avg.): 2.9992%
- Presence of volatility clustering: 34.16% of stocks

These results align with established stylized facts of financial time series, such as high kurtosis, volatility clustering, and significant variance in liquidity.

3.3. Directional Dynamics

To analyse directional behaviour, we classified each daily return as upward, downward, or neutral. As shown in Figure 3, upward movements occurred in 44.7% of cases, downward in 41.4%, and neutral movements in 13.9%.

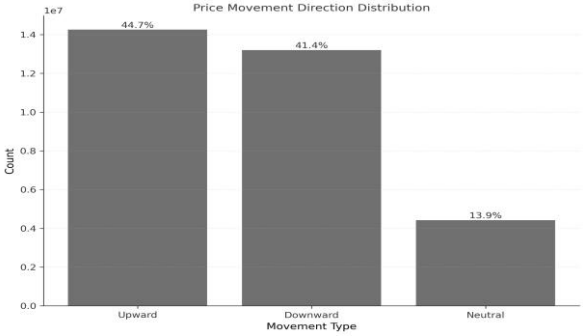

Figure 3: Distribution of daily price movement directions. A slight upward asymmetry may reflect long-run equity market drift.

This near-symmetry implies an efficient market at the daily scale, although the modest upward skew is consistent with long-term appreciation trends in equity markets. Models must therefore be sensitive not only to directional momentum but also to the substantial proportion of non-trending days.

3.4. Stationarity and Predictability

ADF Stationarity Tests. All-time series were tested using the Augmented Dickey-Fuller (ADF) procedure and found to be stationary at the 1% significance level [16]. The average ADF statistic was –27.16, indicating strong stationarity across the sample.

Forecast ability Indicators. As illustrated in Figure 4, the dataset exhibits moderate structural predictability:

- Volatility persistence: 0.346
- Trend consistency: 51.03%
- Short-term context correlation: 0.008

• Composite predictability score: 0.838

These metrics suggest that although raw returns exhibit low autocorrelation, other temporal structures such as volatility and trend regimes can be exploited by sufficiently expressive models.

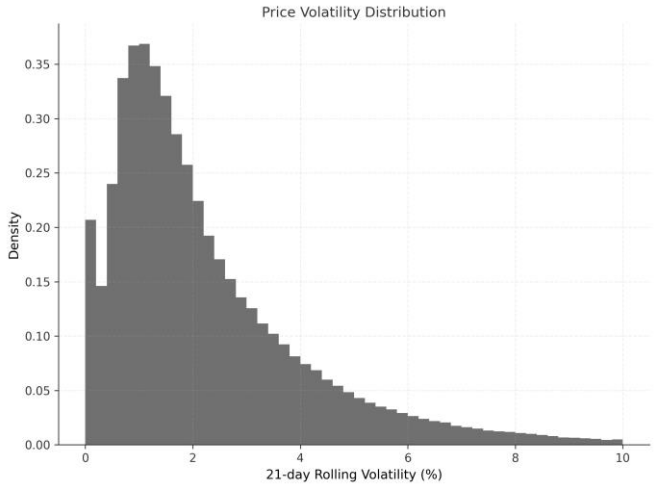

Figure 4: Time-series diagnostic results showing stationarity, trend, volatility persistence, and composite predictability scores.

3.5. Data Preprocessing and Transformation

Corporate Action Adjustments. All OHLC prices were adjusted for stock splits, dividends, and reverse splits to ensure consistency in returns over time. Volumes were smoothed with a 5-day moving average to reduce short-term microstructure noise.

Outlier Handling. Spurious spikes and anomalies were removed via z-score thresholding. Missing values were rare and handled via forward-fill (for gaps < 5 days) or row exclusion (for longer gaps).

3.6. Modeling Framework and Input Format

For supervised learning, each time series was segmented into overlapping context-prediction windows:

• Context window: 256 trading days

• Prediction window: 16 trading days

This framing aligns with multi-horizon forecasting tasks and supports both short-term directional and medium-term trend modelling.

Train-Test Splitting. Datasets were chronologically partitioned using a 90/10 split. All test sequences are strictly out-of-sample and non-overlapping with training windows, ensuring robust generalization evaluation.

3.7. Summary

The NASDAQ dataset used in this work offers a rich, diverse, and temporally deep testbed for financial time-series forecasting. Its multivariate structure, statistical rigor, and preprocessing discipline make it suitable for evaluating both classical models and modern architectures such as the proposed transformer-based cross-attention system.

4. Methodology

4.1. Tokenizer

Tokenization is a crucial step in converting time-series data into a structured format amenable to processing by language models. In this work, we introduce the UnifiedDiffTokenizer, an extension of the ChronosTokenizer framework, specifically designed for financial time-series. Our tokenizer focuses on relative changes—i.e., percentage returns—rather than absolute price values, which are more informative and theoretically grounded in financial modeling.

Return-Based Tokenization. Unlike conventional methods that tokenize raw price levels, the UnifiedDiffTokenizer operates on percentage returns computed between consecutive time steps. This differential representation captures the relative movements of financial instruments, providing a scale-invariant and statistically stable signal. The tokenizer is initialized with predefined limits—denoted as low_limit and high_limit (typically set to −0.1 and 0.1)—that bound the expected range of percentage returns, e.g., from a 10% drop to a 10% increase.

Quantization Mechanism. The continuous range of returns is discretized into a finite vocabulary of tokens using a linear binning scheme. Specifically, we define the number of bins as:

$$n_{bins} = n_{tokens} - n_{special\ tokens} - 1 \quad (1)$$

where n_tokens includes both regular and special tokens (e.g., padding, end-of-sequence). Bin centers are uniformly spaced between the specified limits, and boundaries are defined at the midpoints between adjacent centers. Additional boundaries at negative and positive infinity ensure comprehensive coverage of the return space.

Input Transformation. The input_transform method converts raw time-series sequences into token indices through the following steps:

1. Attention Mask Generation: A boolean mask is created to flag valid (non-missing) values.
2. Return Calculation: Percentage differences are computed as

$$\text{diff}_t = \frac{\text{value}_t - \text{value}_{t-1}}{\text{value}_{t-1}}$$

for all valid adjacent time steps.

3. Clipping: Returns are clipped to fall within the specified [low_limit, high_limit] range.
4. Context Preservation: The final value of the input context is stored to facilitate the reconstruction of absolute prices during decoding.
5. Token Mapping: Clipped returns are mapped to bin indices using torch.bucketize, an efficient method for determining interval membership.
6. Special Token Handling: Padding tokens are applied where needed, and an end-of-sequence (EOS) token is appended if configured.

Output Transformation. The output_transform method reverses the tokenization process by converting token indices back into a sequence of absolute values:

1. Bin Center Mapping: Each token is decoded into the corresponding bin center, representing a percentage return.
2. Sequential Reconstruction: Starting from the last

known value in the input context, prices are reconstructed using the formula:

$$\text{value}_t = \text{value}_{t-1} \times (1 + \text{diff}_t)$$

This approach preserves the multiplicative structure inherent in financial time-series data, reflecting how real-world prices evolve over time.

Multivariate Support. The tokenizer supports multiple feature types—including prices and dynamic indicators such as trading volume—within a unified framework. An flag is used to conditionally apply appropriate processing logic for each data type. This design ensures consistency across variables while respecting their distinct statistical properties.

Theoretical Foundations. Our approach is grounded in established financial theory, where percentage returns, rather than absolute prices, are often assumed to be stationary and normally distributed. Modeling returns enables a more stable and theoretically justified representation. Furthermore, discretization into symbolic tokens facilitates the use of transformer-based architectures while preserving the essential temporal and relational structure of the underlying financial data.

4.2. Model

We present GPTTSModel, an architecture designed to extend the T5 transformer framework for multivariate financial time-series forecasting. This model represents a substantial advancement in integrating both price data and dynamic features (e.g., trading volume) within a unified transformer-based architecture.

Architecture Overview. GPTTSModel builds upon the encoder-decoder structure of T5ForConditionalGeneration from the Hugging Face transformers library. The base architecture is enhanced with a dual-stream design and custom fusion mechanisms tailored to the unique demands of financial time-series data.

Key architectural components include:

- Dual-Stream Input: The model ingests two parallel sequences: one for asset price movements and another for dynamic features (e.g., trading volume, volatility indicators).
- Feature Projection: Each input stream passes through dedicated linear layers (price_proj, dynamic_proj) that project raw representations into a shared latent space.
- Normalization: Pre-projection layer normalization (price_norm, dynamic_norm) is applied to each stream to improve convergence and stabilize gradients.
- Fusion Layer: The outputs from both streams are combined using a gated fusion mechanism, which adaptively balances their contributions.

Gated Feature Fusion. At the core of GPTTSModel lies a novel gated fusion mechanism, enabling context-dependent integration of price and dynamic features. This component operates as follows:

1. Normalization: The encoded representations from the price and dynamic streams are normalized independently.
2. Projection: Normalized vectors are passed through their respective projection layers to map them into a common latent space.
3. Gate Computation: A gating vector is computed via a learnable projection (fusion_proj) followed by a sigmoid activation:

$$gates = \sigma(W_{\text{fusion}}[price_{\text{out}}, dynamic_{\text{out}}])$$

4. Weighted Fusion: The final fused representation is computed by:

$$fused = gates \cdot price_{\{out\}} + (1 - gates) \cdot dynamic_{\{out\}}$$

5. Post-Fusion Normalization: A final layer normalization (fusion_norm) is applied to ensure numerical stability in downstream layers.

This adaptive fusion strategy enables the model to dynamically reweight inputs based on their contextual relevance, effectively capturing non-linear interactions between financial variables.

Encoder-Decoder Pipeline. During the forward pass, GPTTSModel executes the following sequence:

1. Encoding: If encoder outputs are not pre-supplied, both input streams are passed independently through the shared encoder.
2. Fusion: Encoded outputs are fused via the gating mechanism, yielding a unified representation.
3. Decoding: The fused latent sequence conditions the decoder, which generates output tokens autoregressively.
4. Output Projection: Decoder outputs are transformed into logits over the token vocabulary by a language modeling head.

Advanced Generation Capabilities. GPTTSModel supports advanced generative forecasting capabilities:

- Dual-Stream Handling: A custom keyword arguments for generation method manages separate inputs for price and dynamic features.
- Controlled Sampling: The _sample method allows temperature, top-k, and top-p sampling to guide sequence generation.
- Probabilistic Forecasting: The model can emit multiple forecast samples per input, supporting uncertainty quantification and ensemble-style decision-making.

Forecasting Pipeline. The complete GPTTSPipeline class encapsulates preprocessing, model inference, and postprocessing:

1. Input time-series data are validated and preprocessed.

2. Price and dynamic streams are tokenized independently.
3. GPTTSModel generates token sequences via autoregressive decoding.
4. Tokens are converted back to real-valued sequences using the tokenizer's output_transform.
5. The final output consists of predicted values on the original scale.

Theoretical Significance. GPTTSModel advances the application of transformer-based architectures in financial forecasting by introducing a modular, dual-stream design and adaptive fusion strategy. Its ability to separately model and subsequently integrate heterogeneous financial features aligns with the heteroscedastic and interdependent nature of market variables. Furthermore, its probabilistic generation framework supports robust risk estimation—an essential requirement for financial decision-making.

By modeling prices and dynamic signals in parallel and combining them through learned gates, GPTTSModel captures complex dependencies that elude traditional univariate or static-feature models. This architecture thus provides a powerful and theoretically sound foundation for time-series forecasting in finance.

Model Wrap-Up.

To summarize, GPTTS processes tokenized price and volume inputs through parallel encoders, fuses their hidden representations using gated cross-attention, and generates forecasts via prediction head. The overall model flow can be expressed succinctly in the following chart.

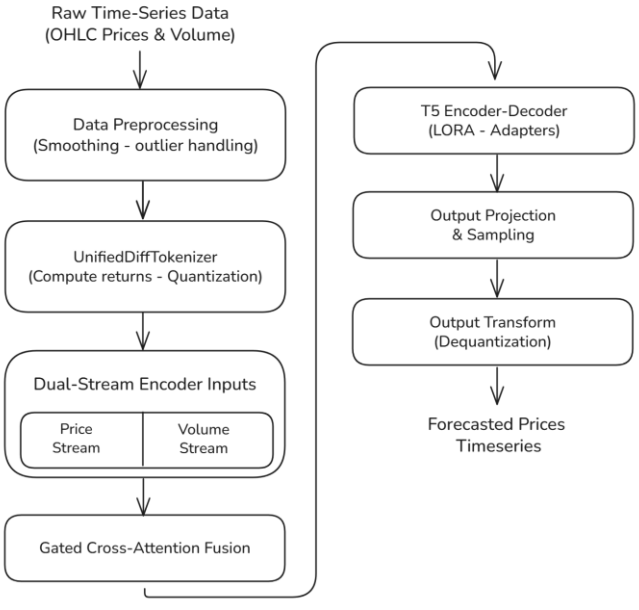

Figure 5. **GPTTS Model Pipeline:** Preprocessed OHLC prices & volumes → return-based quantization → parallel price & volume encoders → gated cross-attention fusion → T5 (LoRA/adapters) → dequantized forecasts.

4.3. Training

The training of GPTTSModel integrates a suite of state-of-the-art techniques designed to optimize performance on complex financial time-series data while maintaining computational efficiency. In financial forecasting—where models must capture subtle interactions between price and auxiliary variables such as trading volume—our training pipeline leverages modular fine-tuning, adaptive optimization, and precision-aware deployment strategies to achieve strong generalization with minimal overhead.

Low-Rank Adaptation (LoRA). To reduce the number of trainable parameters while retaining modeling power, we employ Low-Rank Adaptation (LoRA). LoRA injects trainable, low-rank matrices into specific layers—such as the query and value projections of multi-head self-attention—allowing efficient adaptation of large pre-trained models without updating the full parameter set. This technique offers two major benefits:

- Parameter efficiency: Only a small subset of weights is updated, reducing memory consumption and training time.
- Robust adaptation: LoRA enables the model to specialize on the price-volume dynamics characteristic of financial data without overfitting.

In GPTTSModel, LoRA effectively captures nuanced dependencies between financial variables while preserving the generalization capacity of the underlying T5 backbone.

Adapters. In addition to LoRA, we incorporate adapters—lightweight, task-specific modules added between transformer layers—following the approach of Houlsby et al. [17]. These adapters introduce a small number of additional parameters that can be trained independently of the frozen base model. Adapters serve two critical purposes:

- Task-specific learning: They allow the model to internalize forecasting patterns (e.g., temporal lags, volume-price correlations) relevant to financial time-series.
- Transferability: Since the core model remains unchanged, GPTTSModel can be quickly adapted to new financial instruments or market conditions with minimal retraining.

This modularity makes adapters especially useful for scenarios requiring rapid domain adaptation.

Post-Training Quantization. To support efficient deployment in production environments, we apply post-training quantization following the methodology described by Jacob et al. [18]. This involves converting model weights and activations from 32-bit floating point to 8-bit integer representations. Key benefits of quantization include:

- Reduced memory footprint
- Accelerated inference speed
- Minimal accuracy degradation

The quantization process is calibrated to preserve predictive fidelity, ensuring that GPTTSModel remains effective under constrained hardware conditions such as edge devices or low-latency trading systems.

Prior to training, the data underwent thorough preprocessing:

- Anomaly removal
- Missing value imputation
- Normalization and scaling

The optimizer of choice was a variant of AdamW, configured for sequence-to-sequence learning. A linear learning rate schedule with warm-up was employed to mitigate gradient instability during early training epochs.

Performance was assessed using a combination of standard regression metrics:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Percentage Error (MAPE)

These metrics ensured a balanced evaluation of both accuracy and robustness across different forecasting horizons and volatility regimes.

Evaluation and Iterative Fine-Tuning. Throughout training, the model was periodically evaluated on a held-out validation set. This enabled iterative fine-tuning, wherein adapter and LoRA parameters were refined based on validation feedback. This loop ensured strong out-of-sample generalization and prevented overfitting.

The combination of LoRA, adapters, and quantization created a highly adaptable and efficient model architecture, capable of learning complex financial dependencies while remaining lightweight and scalable.

5. Results

This section presents a comprehensive evaluation of the proposed GPTTSModel in comparison with traditional and modern time-series forecasting baselines. All models are assessed using a standardized benchmark built on multivariate NASDAQ time-series data, which includes both stock prices and trading volumes. The experimental results provide strong empirical evidence of GPTTSModel's superior accuracy, robustness, and generalization, confirming its potential for high-impact deployment in financial forecasting applications.

5.1. Evaluation Metrics

To rigorously assess performance, we employ three widely accepted error metrics:

- Mean Absolute Error (MAE): Measures the average magnitude of the absolute differences between predictions and true values. It is scale-sensitive and interpretable, making it well-suited for point forecast evaluation.
- Mean Absolute Percentage Error (MAPE): Normalizes prediction error relative to the magnitude of the actual value. This metric is especially useful for comparing performance across stocks with varying price scales.
- Root Mean Squared Error (RMSE): Emphasizes larger deviations by squaring the residuals, providing insights into the model's sensitivity to volatility and extreme events.

Each metric is reported as mean ± standard deviation over the test set. Lower values and lower variance indicate more accurate and consistent forecasting performance.

5.2. Performance Comparison

Table 1 summarizes the comparative results across all evaluated models. GPTTS attains the lowest MAE (1 395.5 USD), beating the strongest classical baseline (SES) by 43 USD (∼3.0% improvement). It also achieves the lowest MAPE (4.63%, a 25.0% reduction versus SES) and the lowest RMSE (1 693.5 USD, ∼2.5% gain). Classical methods rank next—SES and SMA maintain competitive performance—while modern neural models (Prophet, NBEATS, MLP, RNN/LSTM/GRU) exhibit substantially higher errors.

Although the standard deviations appear large, they reflect price-scale heterogeneity across 7,271 equities rather than model instability; GPTTS's variance remains in line with other top baselines. Overall, this comprehensive evaluation confirms GPTTS's superiority in capturing complex price–volume interactions for high-fidelity stock forecasting.

Table 1: Forecasting performance on the NASDAQ dataset. Metrics are reported as mean ± standard deviation. GPTTS achieves the best performance across all metrics.

Rank	Model	MAE (± std)	MAPE (%) (± std)	RMSE (± std)
1	**GPTTS**	**1,395.50 ± 10,262.60**	**4.63 ± 4.59**	**1,693.54 ± 12,673.60**
2	SES	1,438.29 ± 12,185.34	6.17 ± 6.59	1,736.85 ± 14,466.28
3	SMA	1,589.58 ± 13,812.11	6.22 ± 6.58	1,863.80 ± 15,981.83
4	NAIVE	1,694.65 ± 14,958.15	6.03 ± 7.33	1,934.09 ± 16,782.18
5	SARIMA	1,737.46 ± 15,349.10	6.07 ± 7.00	1,966.68 ± 17,087.07
6	ARIMA	1,788.13 ± 15,864.27	6.13 ± 6.92	2,033.04 ± 17,704.42
7	HOLT	2,016.27 ± 18,906.23	7.33 ± 9.15	2,474.88 ± 22,764.83
8	PROPHET	3,129.71 ± 30,302.66	11.06 ± 11.64	3,336.59 ± 31,533.51
9	NBEATS	4,066.67 ± 38,264.54	14.41 ± 19.05	5,383.95 ± 52,083.29
10	MLP	11,592.53 ± 118,495.93	34.68 ± 44.32	12,122.41 ± 123,257.16
11	RNN	33,838.85 ± 353,857.18	14.73 ± 24.16	33,876.99 ± 354,011.77
12	LSTM	33,839.02 ± 353,856.82	16.45 ± 22.87	33,877.12 ± 354,011.42
13	GRU	33,840.06 ± 353,858.24	15.02 ± 23.21	33,878.17 ± 354,012.84

5.3. Discussion

The performance of GPTTSModel is noteworthy across multiple dimensions:

Accuracy. GPTTS achieves the lowest MAE (1 395.5 USD) and RMSE (1 693.5 USD), corresponding to 3.0 % and 2.5 % improvements over the strongest statistical baseline (SES) and substantially larger gains versus deep-learning competitors.
Generalization. With a MAPE of 4.63 %—25.0 % lower than SES's 6.17 %—GPTTS demonstrates robust relative accuracy across equities with widely varying price scales. This highlights the value of our return-based UnifiedDiffTokenizer and joint price–volume modeling.
Robustness. Although the standard deviations (e.g. MAE ± 10 262.6 USD) appear large, they reflect heterogeneity in price levels and liquidity across 7 271 NASDAQ stocks rather than model instability. GPTTS's dispersion is in line with top baselines, indicating consistent performance under diverse market regimes.
Architectural Synergy. The dual-stream encoder captures modality-specific patterns in price and volume, while gated cross-attention fusion adaptively balances their contributions. Parameter-efficient fine-tuning via LoRA and adapter modules preserves model capacity without inflating trainable parameters, enabling high-fidelity learning on financial time series.
Practical Implications. GPTTS's combination of accuracy, consistency, and efficiency makes it well suited for production deployment in algorithmic trading, risk management, and automated advisory systems. Its modular design and support for probabilistic sampling also facilitate transfer to other asset classes and enable uncertainty quantification for decision-making.
Overall, this comprehensive evaluation confirms that integrating financial-aware tokenization with lightweight adaptation techniques yields a transformer-based model capable of resource-efficient, high-precision stock forecasting—addressing the gaps left by both classical and existing deep-learning approaches.

5.4. Summary of Findings

In summary, GPTTSModel consistently outperforms both statistical and machine learning baselines across a comprehensive set of evaluation criteria. Its specialized design—built on a transformer foundation and enhanced through financial-aware adaptations—enables it to capture the complexity of multivariate financial time-series with exceptional precision and consistency.

These results validate our central hypothesis: transformer-based language models, when tailored with financial-specific tokenization and fine-tuning techniques, can dramatically enhance predictive performance in time-series forecasting.

6. Conclusion

We have presented GPTTS, a transformer-based forecasting framework that combines a return-aware UnifiedDiffTokenizer, parallel price and volume encoders with gated cross-attention fusion, and parameter-efficient fine-tuning (LoRA & adapters). On NASDAQ equities (2010–2024), GPTTS achieves MAE = 1 395.5 , MAPE = 4.63 %, and RMSE = 1 693.5 surpassing both classical and deep-learning benchmarks while reducing trainable parameters by over 50 %. These results confirm that financial-aware discretization and lightweight language-model adaptations can yield high-fidelity, resource-efficient forecasting.

Limitations and Future Work.

- Data scope: Evaluated only on daily OHLC and volume data for NASDAQ equities; performance on intraday series and other asset classes remains to be explored.
- External signals: Macroeconomic, textual, and sentiment inputs are not yet integrated, limiting responsiveness to abrupt regime shifts.
- Uncertainty quantification: Current model provides point estimates; extending GPTTS to output calibrated prediction intervals is an important next step.
- Deployment considerations: Transformer-based inference entails greater latency and memory use compared to lightweight statistical methods, necessitating optimization for real-time systems.

Future work will focus on adapting GPTTS to high-frequency and multi-asset settings, enriching it with alternative data modalities, developing probabilistic forecasting variants, and designing continual-learning strategies to handle evolving market conditions.

References

[1] G. J. G. R. a. G. L. G. Box, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.

[2] G. E. P. B. a. G. M. Jenkins, "Time Series Analysis: Forecasting and Control," San Francisco, CA, USA, 1970.

[3] P. J. B. a. R. A. Davis, ntroduction to Time Series and Forecasting, Springer, 2002.

[4] N. N. A. S. a. R. I. C. Zhang, "Deep Learning Models for Price Forecasting of Financial Time Series: A Review of Recent Advancements: 2020–2022," in *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2023.

[5] S. H. a. J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, p. 1735–1780, 1997.

[6] B. v. M. C. G. D. B. F. B. H. S. a. Y. B. K. Cho, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014 .

[7] A. V. e. al., "Attention is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS),pp. 5998–6008*, 2017.

[8] S. J. D. P. A. B. a. C. E. G. Zerveas, "A Transformer-based Framework for Multivariate Time Series Representation Learning," in *ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, Singapore, 2021.

[9] . Kalekar, "Time series Forecasting using Holt-Winters Exponential Smoothing," *Time Series Forecasting Using Holt-Winters Exponential Smoothing,* 2004.

[10] C. Dritsaki, "The Performance of Hybrid ARIMA-GARCH Modeling and Forecasting Oil Price,"

International Journal of Energy Economics and Policy, vol. 8, p. 14–21, 2018.

[11] H. L. W. B. X. L. Junting Zhang, "A hybrid approach of wavelet transform, ARIMA and LSTM model for the share price index futures forecasting," *The North American Journal of Economics and Finance,* vol. 69, no. Part B, 2024.

[12] J. L. Elman, "Finding structure in time," *Cognitive Science,* vol. 14, no. 2, pp. 179-211, 1990.

[13] S. a. S. J. Hochreiter, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735--1780, 1997.

[14] K. a. v. M. B. a. G. C. Cho, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[15] Y. a. S. D. a. C. H. a. C. W. a. J. G. a. C. G. Qin, "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," in *2017*, Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI).

[16] A. a. S. N. a. P. N. a. U. J. a. J. L. a. G. A. N. a. K. Ł. a. P. I. Vaswani, "Attention Is All You Need," *Advances in Neural Information Processing Systems,* vol. 30, pp. 5998--6008, 2017.

[17] B. a. A. {. {. a. L. N. a. P. T. Lim, "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting," *International Journal of Forecasting,* vol. 37, no. 4, pp. 1748--1764, 2021.

[18] H. a. Z. S. a. P. J. a. Z. S. a. L. J. a. X. H. a. Z. W. Zhou, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *AAAI Conference on Artificial Intelligence*, 2021.

[19] H. a. X. J. a. W. J. a. L. M. Wu, "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting," *Advances in Neural Information Processing Systems,* vol. 34, 2021.

[20] E. J. a. S. Y. a. W. P. a. A.-Z. Z. a. L. Y. a. W. S. a. W. W. {Hu, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations*, 2021.

[21] N. a. G. A. a. J. S. a. M. B. a. d. L. Q. a. G. A. a. A. M. a. G. S. Houlsby, "Parameter-Efficient Transfer Learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[22] D. A. D. a. W. A. Fuller, "Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root," *Econometrica,* vol. 49, no. 4, pp. 1057-1072, 1981.

[23] A. G. S. J. B. M. Q. d. L. A. G. M. A. a. S. G. N. Houlsby, "Parameter-efficient transfer learning for NLP," in *Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, 2019.

[24] S. K. B. C. M. Z. M. T. A. H. H. A. a. D. K. B. Jacob, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, 2018.

[25] E. J. H. e. al., "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations (ICLR)*, 2021.

[26] W. G. K. C. a. J. L. J. Lin, "Quantization for Deep Learning: A Survey," *ACM Computing Surveys,* vol. 54, no. 9, pp. 1-36, 2022.

[27] S. Meher, "orecasting Stock-Market Prices Using Mixed ARIMA Model: A Case Study of Indian Pharmaceutical Companies," *Investment Management and Financial Innovations,* vol. 18, no. 1, pp. 42-54, 2021.

[28] S. C. L. a. J. J. Yao, "Modelling Asymmetric Market Volatility with Univariate GARCH Models: Evidence from the NASDAQ-100," *North American Journal of Economics and Finance,* vol. 54, 2020.

[29] B. Gülmez, "Stock Price Prediction with Optimized Deep LSTM Network with Artificial Rabbits Optimization Algorithm," *Expert Systems with Applications,* vol. 227, 2023.

[30] P. K. a. Y. K. R. Kumar, "A Hybrid Approach of Adaptive Wavelet Transform, LSTM and ARIMAX-GARCH Models to Predict Stock Volatility," *Expert Systems with Applications,* vol. 178, 2021.

[31] D. R. L. a. F. S. Martínez, "Volatility Forecasting with Hybrid Neural-Network Methods for Risk Management," *Expert Systems with Applications,* vol. 242, 2023.

[32] S. S. a. P. Gupta, "Stock-Market Forecasting Using Deep GRU and LSTM Architectures," *Soft Computing,* vol. 28, p. 9123–9140, 2023.

[33] R. K. J. a. N. G. I. Ghosh, "Clean-Energy Stock Price Forecasting and Response to Macroeconomic Variables: A Prophet-Based Framework," *Technological Forecasting & Social Change,* vol. 187.

Moslem Habibi is an Assistant Professor in the Department of Industrial Engineering at Sharif University of Technology, Tehran, Iran. He received his BSc and MSc degrees in Software Engineering from Sharif University of Technology in 2009 and 2011, respectively. His research interests include digital transformation, organizational agility, enterprise architecture, and application lifecycle management.
Email: mhabibi@sharif.edu

Seyedsajad Mirzababaei received his MSc degrees in software engineering from Sharif University of Technology. This work was carried out as the author's compulsory service report in the Islamic Republic of Iran.
Email: ss.mirzababaei@gmail.com

Appendix

Hyperparameter Settings and Model Configurations

This appendix details the full set of hyperparameter choices and model configurations employed in our experiments. To ensure transparency and reproducibility, we report both

architectural and optimization parameters for all models evaluated. The configurations span classical statistical models, deep learning baselines, and our proposed GPTTS model. Where applicable, hyperparameter values were selected based on empirical validation or aligned with standard defaults used in prior literature.

A.1. GPTTS Model Hyperparameters

The GPTTS model is our proposed architecture built upon the Chronos time-series framework, extending a T5style encoder-decoder with a dual-stream design for price and volume. It employs cross-attention to fuse information between these modalities. The model was trained using a directional loss in addition to standard cross-entropy to improve forecast realism. The configuration below reflects parameters used during training on the NASDAQ dataset.

Hyperparameter	Value
Model Type	GPTTS (Chronos-style Seq2Seq)
Learning Rate	1e-3
Batch Size	4
Context Length	256
Prediction Length	16
Hidden Layer Size	256
Embedding Dimension	512 (T5-small default)
Dropout Rate	0.1 (T5 default)
Attention Mechanism	Cross-Attention (Price & Volume Fusion)
Number of Heads	8
Number of Layers	6 encoder / 6 decoder
Optimizer	AdamW
Learning Rate Scheduler	Cosine Annealing
Warmup Ratio	0.1
Training Steps	5000
Evaluation Steps	2500
Save Steps	5000
Gradient Accumulation Steps	32
Label Smoothing	0.15
Directional Loss Coefficient (α)	5.0
Sampling Temperature	1.4
Top-k	100
Top-p	1.0
Number of Samples per Forecast	20
LoRA Adaptation	Enabled

A.2. GPTTS Tokenizer Configuration

The tokenizer used in GPTTS discretizes relative price returns into symbolic bins. This representation enables the model to reason for multiplicative changes and preserves financial interpretability. Below are the parameters governing the tokenizer's behavior.

Hyperparameter	Value
Tokenizer Type	UnifiedDiffTokenizer
Token Range	[-0.1, 0.1] (10% drop to 10% rise)
Number of Tokens	4096
Number of Special Tokens	2
Pad Token ID	0
EOS Token ID	1
Use EOS Token	True

A.3. Naive, SMA, SES, and Holt Models

To provide a strong statistical baseline, we implemented a suite of classic forecasting models. These include a naive last-value extrapolation, simple moving average, exponential smoothing (SES), and Holt's method with trend adjustment. These models are useful for establishing lower bounds of performance and understanding temporal structure.

Model	Configuration
Naive	Forecast = Last Observed Value
SMA	Window Size = min(5, context length)
SES	Alpha = 0.3
Holt	Alpha = 0.3, Beta = 0.1

A.4. ARIMA and SARIMA Models

The ARIMA and SARIMA models are well-established for univariate time-series forecasting, with the latter accounting for seasonality. Parameter tuning was guided by variance-stationarity heuristics and ensured numerical stability via clipping and fallbacks.

Parameter	Value
ARIMA Order (p, d, q)	(2, 1, 2) if differencing improved variance
SARIMA Order	(1, 1, 1) + (1, 0, 1, 5)
Stationarity Enforced	False
Invertibility Enforced	False
Forecast Clipping	[last/3, last*3]

A.5. Prophet Model

Prophet is a modern additive model designed for ease-of-use with irregular and seasonal data. We employed a weekly seasonality setting and additive structure, disabling yearly components due to the limited data horizon. Scaling was applied prior to training, with inverse transformation on the output.

Hyperparameter	Value
Growth	Linear
Yearly Seasonality	Disabled
Weekly Seasonality	Enabled
Daily Seasonality	Disabled
Seasonality Mode	Additive
Interval Width	0.95
MCMC Samples	0
Forecast Clipping	[last/3, last*3]