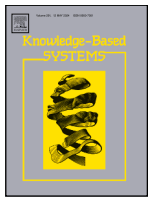
Knowledge-Based Systems 330 (2025) 114449

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Enhancing large language models for bitcoin time series forecasting

Owen Chaffard [a,b,*], Pablo Mollá [c], Marc Cavazza [d], Helmut Prendinger [e]

[a] CardoAI, Milan, Italy
[b] RPTU, Kaiserslautern, Germany
[c] The University of Paris-Saclay, France
[d] The University of Stirling, Stirling, UK
[e] The National Institute of Informatics, Tokyo, 101-8430, Japan

ARTICLE INFO

Keywords:
Time series forecasting
Language models
Financial time series

ABSTRACT

In the recent advancements in application of deep learning to time series forecasting, focus has shifted from training transformers end-to-end to efficiently leveraging the predictive capabilities of Large Language Models (LLMs). Models that encode the time series data to interact with a frozen LLM backbone have been shown to outperform transformers on all benchmark datasets. However, their efficiency on complex datasets, which do not show clear seasonality or trend, remains an open question. In this work, we seek to evaluate the performance of reprogrammed LLMs on the Bitcoin price chart, a financial time series known for its complexity and high volatility. We propose effective methods to improve the performance of Time-LLM, a State-of-the-art (SOTA) method, on such a time series. First, we propose structural improvements to Time-LLM. Second, we suggest an efficient way to handle the non-stationarity of the dataset. Finally, we propose an efficient method for passing additional financial information to the LLM. Our results demonstrate a 50 % improvement on the average percentage loss and a 5 % increase on accuracy of our adapted Time-LLM architecture on Bitcoin data when compared to SOTA models, including the original Time-LLM model. This highlights the impact on forecast accuracy of domain-specific decision making in data processing and feature selection.

1. Introduction

Time series forecasting is a vast domain with numerous real world applications in transportation, healthcare, energy, finance, and many other domains [1]. There has long been interest in the use of various Deep Learning methods for time series modeling, including Convolutional Neural Networks (CNN) [2], Recurrent Neural Networks (RNN) [3] and Graph Neural Networks (GNN) [4]. However, these methods have not been able to consistently outperform other machine learning or statistical methods. Yet, interest in using Deep Learning for time series has been renewed with the appearance of transformers [5]. These models have already revolutionized the domain of Natural Language Processing (NLP), another sequence-to-sequence based modality, and have been applied with varying success to time series analysis. While transformers achieve state of the art (SOTA), their performance for forecasting has recently been put into question, even being outperformed by linear forecasters on certain datasets [6].

The revolution of transformers in NLP has been manifested through the rise of foundation models, i.e., large pretrained models that remarkably are able to adapt to many different tasks using few or even zero-shot transfer learning. These Large Language Models (LLMs), such as LLaMa [7] or GPT-4 [8], have successfully been used across different modalities. It seems that these models are able to perform pattern recognition on sequence modeling beyond their original text-based training corpus. Hence it appears natural to make use of this ability for time series forecasting.

The advantages of using LLMs on time series, rather than transformers, are manifold. First, transfer learning only requires a small number of weights to be updated, and thus is a much more time and data efficient method. This efficiency and ability for zero or few-shot learning is essential for time series as many datasets have rather limited historical data. Second, LLMs benefit from multimodal input and allow the usage of text prompts to guide learning.

Recent advancements in architectural design aim to fully leverage large language models (LLMs) for time series tasks, exemplified by models such as GPT4TS [9], TEMPO [10], and TEST [11]. These models prioritize the *reprogramming* of time series data, a strategy that involves efficiently encoding and embedding temporal information to generate sequences of embedding vectors suitable for LLM prediction. This process facilitates the adaptation of LLMs, originally designed for natural

* Corresponding author.
E-mail addresses: owen.chaffard@cardoai.com (O. Chaffard), pmolla@outlook.es (P. Mollá), marc.cavazza@stir.ac.uk (M. Cavazza), helmut@nii.ac.jp (H. Prendinger).

https://doi.org/10.1016/j.knosys.2025.114449
Received 13 November 2024; Received in revised form 24 August 2025; Accepted 7 September 2025
Available online 11 September 2025

language, to model temporal dependencies and patterns inherent in time series data.

As such each of these models create and train a custom embedder for time sequences to directly interact with the frozen LLM backbone. The best methods for encoding and embedding the time series data yield SOTA benchmark scores on both long term and short term forecasting [12].

In this work, we focus on a financial time series, more specifically on the Bitcoin price chart. Among financial assets, Bitcoin is one of the most attractive for trading due to its propensity for large price action. It is also one of the most challenging, presenting periods of extreme volatility and a high sensitivity to crowd sentiment. It has been shown that it exhibits no simple seasonality [13], a fact that sets it apart from the benchmark datasets most researchers use, which are often highly seasonal. Bitcoin is a relatively new asset class and thus has limited historical data while being highly non-stationary. Our evaluation of LLM-based methods on the Bitcoin dataset is aimed at showcasing their abilities beyond simple seasonal datasets.

We have singled out the Time-LLM architecture proposed by Jin et al. [12] as the basis for our study. It is one of the most recent and best performing architectures using LLMs, outperforming earlier models like GPT4TS [9]. The TEMPO architecture [10] was not considered even though its performance on benchmark datasets is similar. The reliance of TEMPO on a simple seasonal decomposition of the signal makes it suboptimal for use on the Bitcoin dataset.

Our work proposes improvements to the Time-LLM model inspired by the challenges of forecasting Bitcoin price.

Non-stationarity: The Bitcoin price series is non-stationary [14]. Since traditional time series methods such as differencing are insufficient to address the non-stationarity, we discuss and compare more advanced methods such as fractional differencing [15] and Reversible Instance Normalisation (RevIN) [16].

Sparse information: Due to limited historical data of Bitcoin (it became active only around 2017) and its complex price action, using only price data is insufficient for achieving good forecasting performance of the model.

To address the issue of limited information, our work focuses on finding an efficient way of making use of multivariate capabilities of LLM-based models. First, we implement a method that was introduced for usage with a transformer based model called the “inverted transformer” [17]. We show that this method of embedding time series data is (1) superior to standard patching and channel independence [18] and (2) computationally more efficient. Second, we build on the prompt-as-prefix method introduced by Lin et al. [12] to efficiently add new variables to our input time series. Specifically, we implement basic technical analysis [19] (automated trend lines, and other technical indicators) as input to our model.

Overall, the main contribution of our work is two-fold:

- We propose structural improvements, such as a more suitable time series embedding method, to the Time-LLM architecture. This leads to better performance and efficiency,
- We report a case study on a real-world, challenging, time series, which involved advanced techniques for addressing non-stationarity as well as exploiting prompting techniques to incorporate specific but formalized knowledge, such as chart information.

The remainder of the paper is organized as follows. Section 2 provides an overview of the state of the art using transformer models and large language models for time series forecasting. Section 3 describes our improvements of the Time-LLM architecture. Section 4 motivates fractional differencing as an effective method to address the non-stationarity of the Bitcoin time series. Section 5 compares the performance of our *basic* forecasting model, i.e. the model that only uses basic chart data, to other state of the art models. Section 6 explains additional information relevant to financial forecasting, including technical analysis. Then, in Section 7 we present the effect of adding such information on the performance of our model, called the *extended* forecasting model. Finally, Section 8 synthesizes our findings and suggests future work.

2. Related works

Our analysis of previous works is centered on recent approaches involving transformers and foundation models. We first review the use of transformers, then foundation models both LLM and time series models. Finally, we report on the limited work on applying these techniques to financial time series analysis.

2.1. Transformer models adapted to time series data

Recent developments in the study of time series came with the arrival of the transformer architecture [5]. This architecture allowed for much faster training and better performance for long term forecasting. Architectures adapting the base transformer model to enhance performance on time series data have been created, such as Informer [20], Autoformer [21] or Reformer [22]. However, the performance of fully trained transformer based models has also been criticized; on some datasets, transformer models have even been outperformed by linear models [6].

Mai et al. [23] studied Transformer, Informer and Autoformer models using the hourly Bitcoin dataset, but longer input sequences could not improve performance. On the other hand, the introduction of patching and channel independence in the PatchTST model [18] is a different way to embed a time series input sequence, which can make better use of longer input lengths. Patching also introduced more semantic meaning into time series embedding. Here each aggregate of points in time represents a potential pattern, comparable to a word in natural language.

More recently, Liu et al. [17] proposed the inverted transformer architecture. This new embedding approach was shown superior to patching. By using the whole input sequence as a single to-be-embedded aggregate, this architecture supports arbitrarily large input sequences. It also processes the different variables of the time series in parallel, unlike channel independence [18], which processes them as different samples in each batch. In our work, we will apply a similar embedding method to the inverted transformer, and show its superiority to patching even for interaction with frozen LLM backbones.

2.2. Foundation models for time series forecasting

The current SOTA in deep learning based time series modeling consists of making use of the vast potential of pretrained large language (foundation) models such as LLaMa2 [7] or GPT4 [8]. Some works feed temporal data directly into the LLM, such as promptcast [24] or LLM-Time [25]. These methods support the use of top LLMs such as GPT4 but are limited to zero-shot forecasting, because access to the model is restricted.

Another method that emerged recently consists of pretraining foundation models directly on time series data rather than text. The most advanced methods include the Chronos family of models [26] or the MOIRAI model [27]. However, the heterogeneity of time series data results in difficulties creating a coherent training corpus, and forces the application of major restrictions on the models. Notably, these foundation models often do not permit the use of multivariate time series. Therefore, we have chosen not to explore this method.

The most widely used method, and the one leading to the best results, is to directly use a frozen LLM backbone and train an embedder adapted to the modality of time series. Chang et al. [9] first introduced this method using a linear embedder in combination with patching to feed time series data to a fine-tuned GPT-2 backbone. More recent approaches make use of the open source LLaMa-2 model, which is a more refined and larger LLM. Instead of fine-tuning the LLM, the

focus is set on finding efficient ways to adapt the time series data to activate LLaMa's full pattern recognition ability. Cao et al. [10] achieve good results by using standard seasonal decomposition of the signal before patching and linear embedding. They show that this type of decomposition is hard for the LLM to learn on its own, as the attention layer is similar to a PCA and seasonal and trend components are not orthogonal.

Financial assets and Bitcoin in particular are different from the typical time series used in benchmark data sets in that they do not exhibit simple seasonality. Thus, the method chosen as the basis of our work is the Time-LLM architecture introduced by Jin et al. [12]. Besides patching, this model uses a cross attention layer with a pool of text prototypes based on the LLM vocabulary (pool of word tokens used for LLM pre-training). This patch reprogramming method supports a better alignment between the language modality and time series data. The embedded vectors are projected onto those of words that are familiar to the LLM. Additionally, Time-LLM introduced the prompt-as-prefix method to provide information about the dataset and details about the task, which activates the LLM's abilities. The prompt can be used to provide explanations on multiple variables and guide the model.

2.3. Financial time series analysis based on LLMs

Publicly available work on time series forecasting using Deep Learning for financial time series has been quite limited. The lack of simple seasonality, among other challenges, might have discouraged researchers to include these time series in the benchmark datasets. Most research specifically concerning financial data has been done using Chat-GPT and has focused on text based evaluation of assets rather than standard forecasting [28]. Mai et al. [23] conducted a preliminary evaluation of the performance of transformer based models on the Bitcoin dataset with some promising results, based on the hourly chart. Our research builds on these previous attempts to show that a reprogrammed LLM can achieve worthy results on actual forecasting for the Bitcoin price chart.

3. Forecasting model

In this section we will focus on the model architecture itself. The complete framework used in this work is presented in Fig. 1. Our architecture is partially based on the Time-LLM model, from which it differs in two major ways :

- The patching method of Time-LLM is replaced by an embedding method introduced for the "inverted transformer" [17],
- The dimensionality reduction method used in Time-LLM is replaced by Principal Component Analysis (PCA).

We also devise a different normalization method, which will be discussed in Section 4.

3.1. Time series embedding

Our proposed model differs from Time-LLM in the way it embeds time series into an LLM embedding space. The baseline model employs the patching strategy of PatchTST [18], which segments each input sequence into fixed-length *patches* that are embedded separately, yielding a sequence of vectors (Fig. 2). Multiple variables are processed under channel independence: covariates are treated as additional samples of the main variable, and multivariate learning occurs only via weight sharing. Fig. 2 highlights how this design presents three limitations for our setting:

- A fixed patch size may not align with meaningful temporal patterns of varying size and frequency.
- Channel independence impedes the model's learning when treating variables differing in dynamics, scale, and bounds (e.g., smoother moving averages or bounded indices such as the RSI) as homogeneous samples.
- The lack of direct multivariate modeling limits the impact of the LLM prior knowledge of technical indicators.

To address these issues, we adopt the inverted embedding of the iTransformer [17]. Each input sequence is encoded as a single

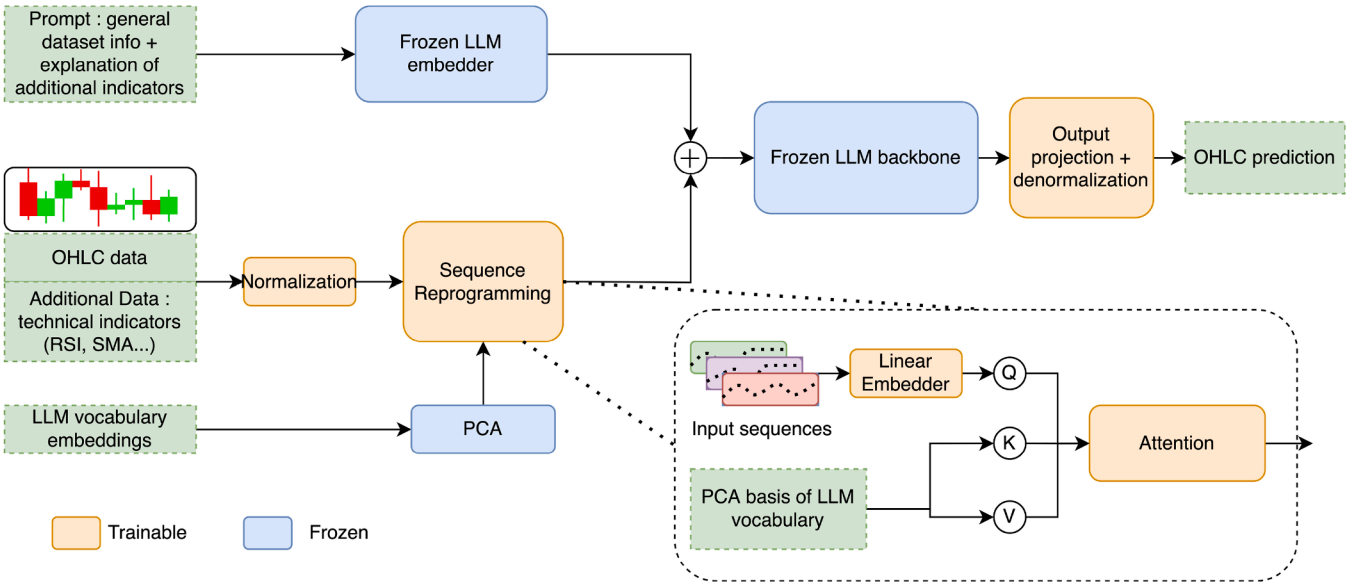


Fig. 1. The proposed framework for using LLMs for Bitcoin time series forecasting, based on the Time-LLM architecture [12]. The price chart is augmented with additional data derived from technical analysis (Section 6). It is then normalized (Section 4) and linearly embedded (Section 3.1). The vector embeddings of the input data are aligned with prior knowledge of the LLM by being fed through a cross attention layer whose keys and values are principal components of the LLM's vocabulary (Section 3.2). The obtained realigned embeddings are then concatenated with a prompt that serves to give general information about the dataset, and to provide explanation for the technical analysis used to augment the dataset (Section 7.1). These vectors are fed through the frozen LLM backbone and projected onto the desired output.

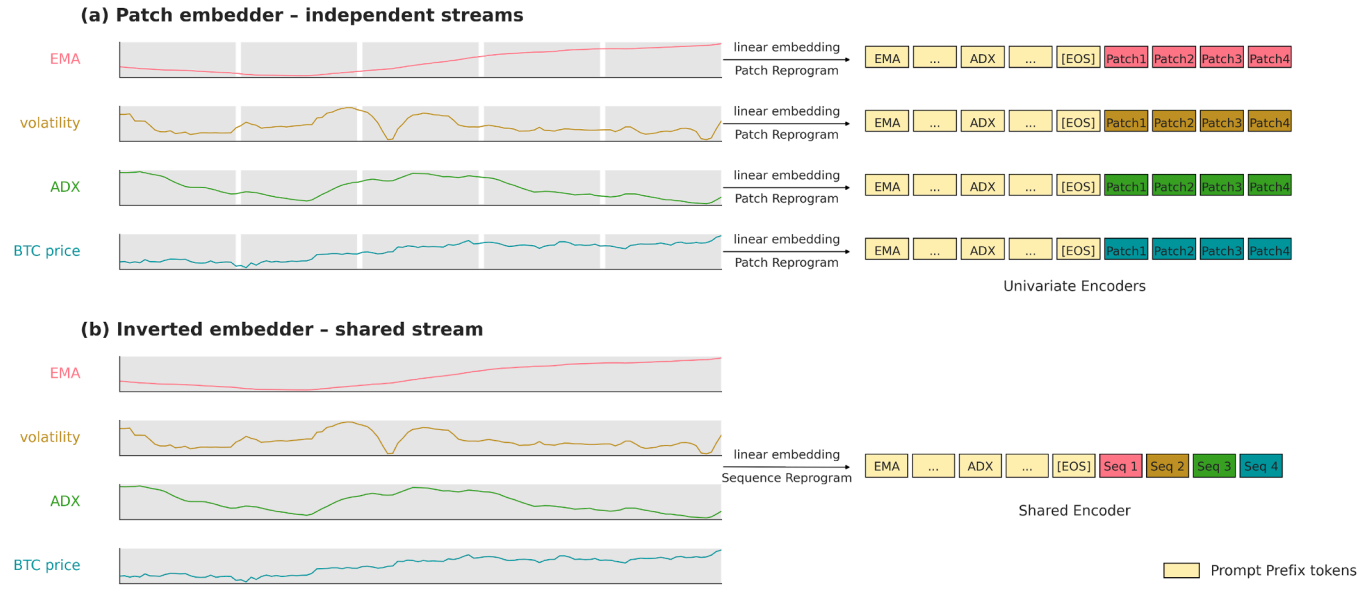


Fig. 2. Visualisation of the two input-embedding strategies used to feed multivariate time series into an LLM backbone. **(a)** *Patch embedding:* Each time series feature (coloured lines) is cut into fixed-length windows ("patches", grey shading). These patches are embedded and prepended with a textual prompt to form the input sequences. Under the channel independence assumption, each univariate stream is independent and multivariate learning only occurs implicitly through weight sharing. **(b)** *Inverted sequence embedding:* The full trajectory of each feature is embedded once, yielding a single token that follows a common prompt. In this configuration the LLM backbone applies attention to the variate token sequence, allowing for inherently multivariate computation.

high-dimensional vector to preserve sequence-level semantics, while attention is applied across variates (variate attention) to model explicit multivariate dependencies. This is particularly pertinent for forecasting BTC price from technical indicators, where interactions among moving averages, indices, and volatility measures all inform price dynamics. This further enhances the impact of the prompt prefix which contextualizes the covariates and leverages the LLM's prior knowledge of technical chart analysis through chain-of-thought. Overall, this design helps capture meaningful covariance structures and makes more effective use of multivariate information. Fig. 2 highlights in particular the efficiency of the design, which allows the use of arbitrarily long input sequences.

3.2. Sequence reprogramming

After embedding time series data into the latent vector space of a large language model (LLM), each sequence-represented as a single vector-is subjected to a *sequence reprogramming* layer. The goal of this layer is to align the embedded time series vectors with the LLM's textual vocabulary vectors, thereby fully leveraging the model's reasoning capabilities. In order to achieve this, we employ a cross-attention mechanism in which the time series vectors serve as queries, while the vocabulary embeddings act as keys and values.

As a result, time series vectors are projected onto the subspace spanned by the LLM's vocabulary. However, since the vocabulary size in models such as Llama exceeds 32,000 tokens, direct cross-attention is computationally prohibitive due to the resulting large parameter count. Thus, dimensionality reduction is essential for efficiency.

Within the Time-LLM architecture, this challenge is addressed by inserting a linear layer that compresses the vocabulary embeddings into 1000 latent text vectors, referred to as text prototypes. This approach, however, introduces additional learnable parameters and requires the dimensionality reduction mapping itself to be optimized during training, which may hamper efficiency and slow convergence.

To overcome these limitations, we instead utilize principal component analysis (PCA) for dimensionality reduction. PCA can be done offline prior to training, and thus does not introduce additional trainable parameters. It also guarantees optimal preservation of vocabulary information with respect to the L2 norm. It is well established that linear dimensionality reduction layers ultimately converge to the PCA basis [29], further supporting the idea that PCA is optimal as a means to cover the LLM reasoning space in the most efficient way.

Table 1
Ablation study of our structural changes to the Time-LLM architecture. The percentage improvement over Time-LLM are expressed for the SMAPE and the average Time-per-Batch during training.

Model	SMAPE	Time-per-batch
Time-LLM	–	–
+ *inverted embedder*	28.3 %	42.9 %
+ *PCA*	30.1 %	47.5 %

3.3. Prompting

Once reprogrammed, the time series vectors are concatenated with an embedded textual prompt. We use the prompt-as-prefix method from the original Time-LLM model (see Fig. 2), which allows us to inform the model about the dataset during training.

In our scenario, prompts specifically serve to introduce additional data and information specific to financial TS such as technical indicators. We seek to activate the model's prior knowledge of these indicators that were likely encountered in its training corpus (Fig. 5 shows our basic prompt structure).

Overall our architectural changes to TimeLLM result in a considerable empirical improvement in both model forecast accuracy and in training efficiency (measured as the average time-per-batch during training). Table 1, showcases this through an ablation study, in which models were compared with identical hyperparameters, including the choice of LLM–Our best reported backbone in LLama2-13B–and input sequence length, which was limited to 50 days–compared to our best model using an input of 200 days–due to memory limitations with the TimeLLM model. All models were also trained using the same prompt (our reported basic prompt).

4. Handling non-stationarity

One of the essential challenging aspects of time series, especially when using machine learning, is dealing with non-stationarity. Indeed most datasets are non-stationary, while neural networks operate under stationarity assumptions. This is particularly problematic when forecasting financial assets and in particular Bitcoin, which shows a heavily varying average price, and even high variation in its volatility. Shifts of Bitcoin can be very sudden, with price change of the order of 10 % often happening over a single day.

Since stationarity is essential for machine learning, detrending and normalization have been a central part in the study of time series. This has been traditionally done using the standard seasonal decomposition, which allows the separation of the trend and a normalized seasonal and residual part [30]. Using this decomposition leads to extremely good results especially on datasets with strong seasonality, as exemplified by the success of the TEMPO architecture [10].

However, financial time series cannot be captured well by this sort of decomposition. They often do not show specific seasonality, and additionally have very strong stochastic noise; random fluctuations or residuals can often be of the order of the trend. In this context, we have identified two methods to deal with non-stationarity in the search for one that would be compatible with our Time-LLM approach.

4.1. Reversible instance normalization

A newer method for dealing with non-stationarity was introduced by Kim et al. [16] in 2022. This method is called "reversible instance normalization" (RevIN) and has already been widely adopted with both transformer and LLM-based forecasters [9,18]. RevIN circumvents the problem of globally detrending the data and instead uses instance normalization. Each batch is normalized before being passed through the model and de-normalized after the output projection. This effectively removes the non-stationary statistical information from the time series while it is being passed through the model.

One of the issues with this approach is dealing with shifts in these statistical values in the input. For example, a sudden uptrend could mean the last value is not aligned with the average that will be added to the output. To address this problem, the RevIN method adds a trainable affine transformation to the denormalization. The learned biases can correct for distributional shifts in the time series.

The RevIN method seems to be the most desirable at first glance. Indeed, when applied to the Bitcoin price chart, this method yields the best results in terms of the average percentage error on the prediction for short horizons. However, this method presents limitations, notably when the price action becomes too significant, which we demonstrate by studying the performance for 14-days-ahead prediction (see Section 4.3).

Indeed, we find that the variance of the predicted price action is much lower than that of the dataset, and thus the model is enable to learn an accurate approximation of the BTC close price distribution. We believe the denormalisation and affine transformation hinder the model's ability to learn more extreme predictions.

To solve this issue we use preprocessing on the dataset to render the time series stationary.

4.2. Fractional differencing

The method traditionally used to detrend financial data is differencing. The use of logarithmic returns further renders data completely stationary and bounded, removing the distribution shifts in volatility. One particular issue of differencing is that it negates long-term effects and dependencies. Indeed the differenced signal at time t only depends on the previous step. While this may be suitable for time series whose dependencies are predominantly short-term—such as signals adequately described by ARIMA processes—the importance of long memory effects in cryptocurrency prices have been highlighted in the literature [31].

Fractional differencing offers a compromise between leaving a series untouched (thus retaining non-stationary long-range dependencies) and applying an integer first difference that erases long-term dependency. The fractional difference of order $d \in [0, 1)$ is an infinite series of lagged differences with power-law decaying weights. Fig. 3(a) visualizes the decay of the absolute value of lag weights for representative values of d. While $d = 1$ yields an abrupt cut-off after a single lag, smaller fractional orders allocate non-negligible weight to observations tens of periods in the past, thereby preserving information that would otherwise be discarded. This method was first introduced by Granger et al. [32] alongside the statistical ARFIMA models for long memory time series forecasting.

Since then, this method has been advocated for as a pre-processing step for machine learning based forecasting in finance [33]. Its superior performance over regular differencing in financial data was empirically demonstrated in time series of stock indexes [34].

Nevertheless, its adoption remains sporadic, and, to the best of our knowledge, no prior work has combined it with state-of-the-art models for time-series prediction.

4.3. Preprocessing pipeline and empirical comparison

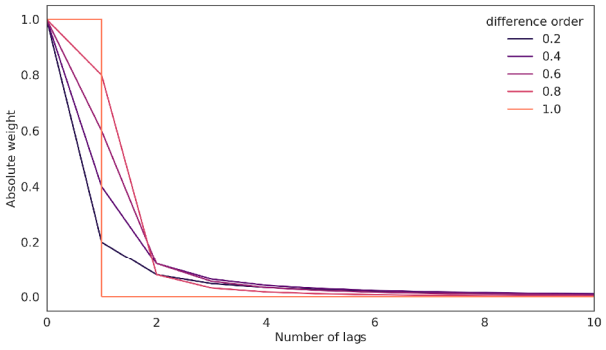
Our preprocessing pipeline selects the optimal differencing order as the lowest value for which the unit-root null hypothesis of the Augmented Dickey-Fuller (ADF) stationarity test is rejected [35]. This approach maximizes memory retention in the time series while ensuring stationarity. To avoid data leakage, the entire search for the optimal order is conducted exclusively on in-sample (training set) data. As illustrated in Fig. 3(b), we plot the ADF p-value versus differencing order and determine that a fractional difference of $d^\star = 0.52$ suffices for achieving stationarity at the 5 % threshold for the BTC close price. The Pearson correlation with the raw series, shown as a function of d, demonstrates that this order retains 77 % of the original linear correlation, while integer differencing ($d = 1$) preserves merely 2 %, confirming that full differencing erases nearly all long-horizon structure. Fig. 3(c) visualizes the transformed series' trajectories, where fractional orders are shown to attenuate slow trends while preserving overall dynamics.

Fig. 4 presents a comparison between the RevIN and fractional differencing (fracdiff) preprocessing methods in the context of 14-day forecast models. The out-of-sample distribution of model predictions is examined in Fig. 4(a) and (b). Fig. 4(a), employing Kernel Density Estimation, reveals that RevIN's predicted returns are concentrated near zero with few predictions exceeding 7 % in absolute change, whereas fractionally differenced predictions more closely align with the empirical distribution, albeit with a slight upward bias.

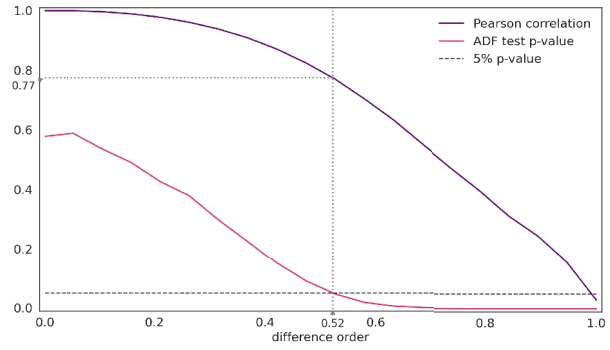
In Fig. 4(b), the differences in distributional fit are quantified using Kolmogorov-Smirnov (KS) statistics and Kullback-Leibler (KL) divergence. The fractionally differenced model's KS statistic is over 50 % lower than that of RevIN, and although both models are rejected by the KS test, fractional differencing achieves a substantially higher p-value ($p_{\text{fracdiff}} = 0.002$ vs. $p_{\text{revin}} < 10^{-10}$). KL divergence analysis further underscores fractional differencing's superiority, with KL integrand peaks indicating RevIN's failure to account for frequent large BTC price moves in the 7 % – 15 % range. Fig. 4(c) provides a qualitative example in a high-volatility period; fractional differencing captures rapid dynamics, whereas RevIN predictions remain unresponsive to volatility shifts.

Given that distributional similarity alone may obscure temporal misalignment, Fig. 4(d) displays return forecasts sequentially against true values. The fracdiff model not only maintains a higher variance aligned with BTC returns but also achieves a superior 56 % directional accuracy for 14-day horizons.

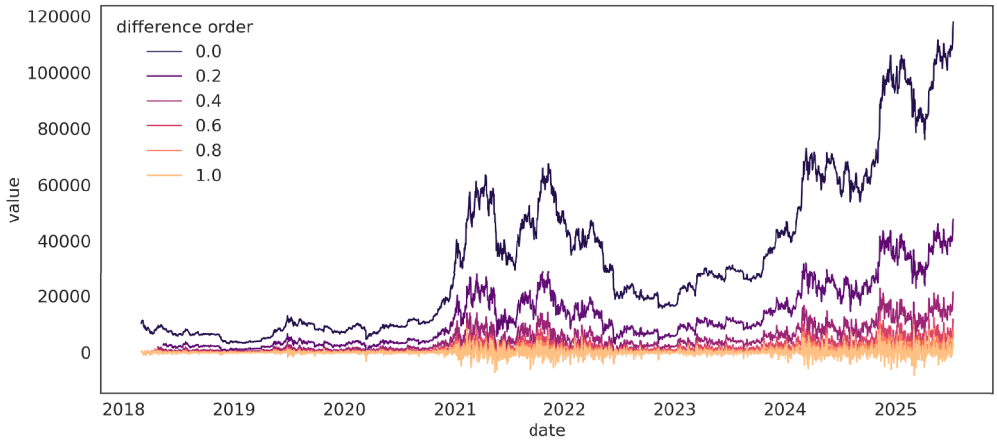
In summary, RevIN's reliance on affine transformations appears to induce static predictions under significant regime shifts, concentrating predictive information in the bias term rather than in the model forecast.



(a) : Absolute value of the lag coefficients for various fractional differencing orders ($d \in (0, 1]$)



(b) : Plot of the Pearson correlation with the original time series and ADF stationarity test p value by fractional differencing order. The selected order (p value of 5%) is highlighted.



(c) : Transformed Bitcoin Price chart for different fractional differencing orders $d \in (0, 1]$

Fig. 3. Detailed study of the effect of fractional differencing on the long-term memory of the Bitcoin price chart.

In contrast, fractional differencing achieves stationarity with minimal information loss, yielding predictions with realistic variability that better match both the temporal and distributional properties of BTC price returns.

5. Results for basic forecasting model

In this section, we first introduce the data used for our experiments and provide some details on the training process. Our benchmarking highlights the advantage of our architecture compared to Time-LLM and other SOTA models. We call our forecasting model 'basic' as it only considers price data and volume.

5.1. Data collection and preprocessing

Our experimental setup utilizes daily candlestick data, comprised of open, high, low, and close (OHLC) prices, aggregated from the Binance and Bitstamp exchanges. The collected data covers the Bitcoin market from the start of 2018 to mitigate the effects of earlier price movements, which were predominantly influenced by retail participation and consequently exhibited high unpredictability.

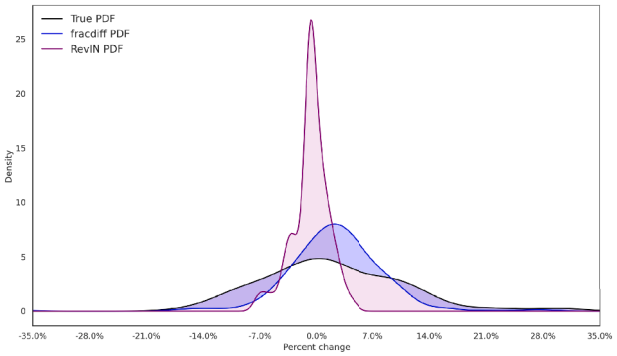
The price was computed as the volume-weighted average of both central exchanges in order to accurately reflect the market state and smooth out possible artifacts in each exchanges data due to low volume trading days. The overall daily trading volume was also incorporated as a feature which provides supplementary insights into market activity beyond price dynamics. Employing daily frequency data appears as a good compromise between maintaining a high signal-to-noise ratio and ensuring sufficient samples are collected for training deep learning models.

The final dataset comprised approximately 3000 instances, partitioned using an 85-15-15 % split for training, validation, and test sets, respectively. The training segment spanned from early 2018 to 2023, followed by validation on 2023–2024 data, and testing from mid 2024 to mid 2025.

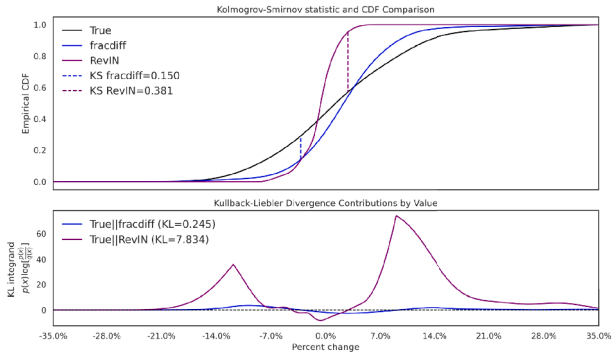
While the number of training samples may appear low for typical Deep Neural Network training, it aligns well with LLMs, which, due to their relatively modest number of trainable parameters in this context, can effectively operate on shorter time series. As outlined in Section 4.3, the preprocessing workflow applies fractional differencing to the OHLC data. During training, loss is computed on the differenced data, while inference and evaluation involve reconstructing the original price series via inversion of the fractional differencing process.

5.2. Training

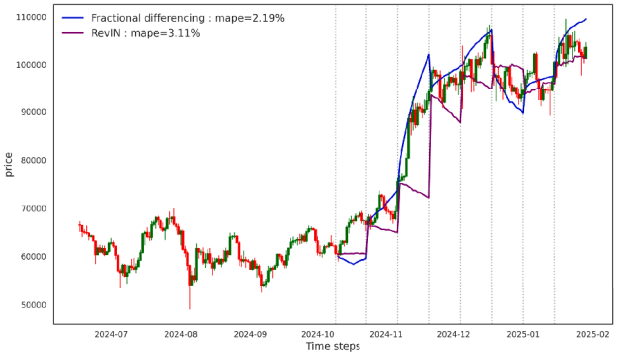
As described in Section 3, our architecture embeds time series sequences into high-dimensional vectors, concatenates them with prompt vectors, and inputs the result into a frozen large language model (LLM). We found that using 200 PCA components of the LLM vocabulary in the reprogramming layer was sufficient in order to obtain optimal results. OUr memory efficient architecture allows use to use larger LLMs. Indeed, while the original Time-LLM work utilizes both GPT-2 and LLama2-7B, our experiments indicate that LLama2-13B yields superior



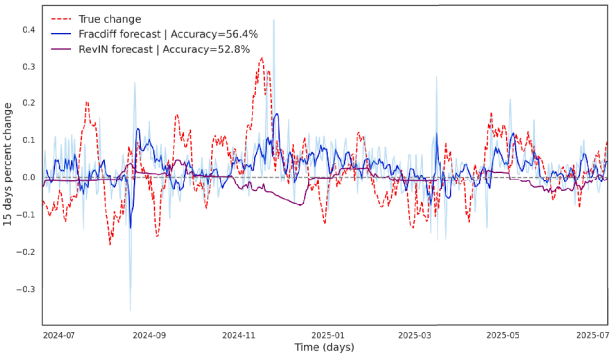
(a) : Kernel density estimation of the predicted and true pdfs of percent change after 14 days



(b) : Comparison between the true and forecast distribution through Kolmogorov-Smirnov statistic and Kullback-Leibler cross-entropy.



(c) : RevIN and fracdiff models' forecast on out-of-sample data (legend indicates the average percentage error of each model).



(d) : Temporal distribution of forecast and true 14 days-ahead percent change along the testing dataset.

Fig. 4. Detailed comparison of RevIN and fractional differencing detrending for the performance of 14-days ahead forecasting.

Simple Prompt

This dataset is the Bitcoin daily price chart.
Below is the information about the input time series:

[Domain]: Bitcoin is the leading cryptocurrency. The bitcoin price is a highly volatile price chart which is globally on an upward trend although it oscillates between bull and bear market cycles that last around 1 to 2 years.
[Data]: Each data point indicates the opening, high, low and closing price of Bitcoin as well as the volume.
[Instructions]: Predict the detrended data for the next *pred_len* steps given the previous *seq_len* steps.
[Statistics]: The input has a minimum value of *minprice* and a maximum value of *maxprice*, with an average value of *meanval*.

Fig. 5. The simple prompt used in our experiment. The prompt structure is derived from the one introduced for Time-LLM.

performance [7]. We attribute this improvement to the model's larger embedding dimension, which is advantageous given our approach of embedding the entire sequence simultaneously. Specifically, LLama2-13B features $d_{model} = 5120$, compared to 4096 for LLama2-7B and 768 for GPT-2. We use the simple prompt illustrated in Fig. 5. The input sequence length is 200, with a prediction horizon of 7 days, providing an appropriate window for evaluating short-term forecasting accuracy.

Our implementation utilizes PyTorch Lightning to facilitate efficient experimentation and model management. During each forward pass, output embeddings are generated, after which components associated with prompts and auxiliary data are excluded. The remaining outputs are projected onto the desired forecast targets via a linear layer. Although closing price prediction is our primary objective, we observe that calculating loss over the complete OHLC (Open, High, Low, Close) data allows the model to leverage the full complexity of candlestick charts. We employ the Mean Average Percentage Error (MAPE) loss, which is advantageous due to its scale invariance. Model optimization is performed using stochastic gradient descent with the Adam optimizer, and the learning rate is selected via PyTorch Lightning's built-in learning rate tuner. To mitigate overfitting, early stopping based on validation set performance is incorporated into the training process.

5.3. Comparison to SOTA methods

Our model architecture was compared against some of the SOTA methods on the specific financial instrument of Bitcoin, forecasting 7 days ahead. The following metrics are used:

- SMAPE (Symmetric mean absolute percentage error) measures the accuracy of the forecast by calculating the percentage error between predicted and actual values (closing prices of Bitcoin),
- RMSE (Root mean squared error) provides an idea of the magnitude of the error in the same unit as the original data (USD in case of Bitcoin),
- Accuracy measures the ability of the model to correctly assess the direction of price movement.

SOTA methods include GPT4TS [9] and Time-LLM [12] that use reprogrammed LLMs, and PatchTST [18], which introduced patching as a way of encoding time series applied to a regular transformer.

We can see that our model outperforms other models, including Time-LLM, which served as the basis for this work's architecture, with

Table 2
Comparison of model performance on forecasting 7 days for Bitcoin closing price.

Method	SMAPE	RMSE	Accuracy (%)
DLinear [6]	9.19	3029	50.7
GPT4TS [9]	7.72	2873	49.1
PatchTST [18]	7.55	2384	48.9
Time-LLM [12]	7.40	2333	49.4
Our Basic Model (based on [12])	**3.42**	**1469**	**52**

a 50 % improvement on the average percentage loss and a 5 % increase on accuracy (see Table 2).

Our model is not only the best performing on the Bitcoin dataset, but also the most susceptible to being improved by additional data. This will be discussed in the next section.

6. Technical chart information

One of the most important aspects of financial forecasting is the use of technical analysis [19]. The consideration of technical indicators and trend lines, as well as external information such as market sentiment, allows traders to identify setups where price changes can be predicted with a certain probability. Our architecture is designed to efficiently leverage the information contained within this additional data.

6.1. Technical analysis

6.1.1. Technical indicators

As a first addition to our basic model we decided to focus on widely used indicators, such as exponential moving average (EMA), relative strength index (RSI), and volatility. These indicators give more context to the price action.

- Exponential moving average (EMA) of 50 days filters out the fluctuations and gives a view of the global trend,
- Volatility is computed as an average of the variance of the price, giving information of the strength of expected short term price action,
- RSI is an index on a scale of 0 to 100 that gives an idea of how strong or weak the asset is: an RSI lower than 30 suggests the asset is oversold and an RSI higher than 70 suggests that the asset is overbought. In either case, a trend reversal can be expected. However, the RSI (unlike support or resistance trend lines) does not help to determine when the reversal will happen.

Additionally, one of the issues we notice in model prediction using the Bitcoin dataset is its tendency to only predict trend continuation. Thus we chose to implement the average directional index (ADX) to help with trend detection.

- ADX gives a value in [0,100] indicating the strength of the underlying trend. When the ADX value is above 25, the underlying trend can be considered strong, whereas values below 20 indicate periods of ranging where no direction is particularly favored.

We will use the prompt as an explanation of the index to instruct the model to look for trend continuation when the ADX is high and for reversal/ranging when it is low (see Section 7.1).

6.1.2. Support and resistance (SR) trend lines

To further guide model prediction, we implemented an automated trend line algorithm. This algorithm allows us to draw support and resistance lines based on recent price action (see Fig. 6). We have found the best way to pass this information to the model is to add the normalized distance to the support and resistance trend lines to our data set.

In the prompt we specify that reversals should be expected when the price is close to a support or resistance trend line (see Section 7.1).

6.1.3. Market sentiment

Analysis of social behavior is very helpful for predicting price movement. As evidence of its relevance, we added the 'fear and greed' index to the dataset. This index compiles social media and other related data to provide an estimation of the market sentiment. It provides a

Fig. 6. Automatically computed support and resistance lines of a wedge pattern, showing the subsequent breaking out of the price. The highlighted points (local extrema of order 10) show that trend reversal is more common the closer the price is to the support or resistance line.

Extended Prompt

This dataset is the Bitcoin daily price chart.
Below is the information about the input time series:

[Domain]: Bitcoin is the leading cryptocurrency. The bitcoin price is a highly volatile price chart which is globally on an upward trend although it oscillates between bull and bear market cycles that last around 1 to 2 years.
[Data]: Each data point indicates the opening, high, low and closing price of Bitcoin as well as the volume. Exponential moving average, volatility and relative strength index are also indicated. The trend strength index is also given. A high value (above 25) indicates a strong upward or downward trend. In this case the price is likely to keep following the trend. At low values (below 20) of this index, the price is ranging, and trend reversals are more likely. During the periods of ranging, trend lines are useful to pinpoint trend reversals.
The 'fear and greed' index is also given, this index provides a view on the overall sentiment surrounding bitcoin. Its value ranges between 0 to 100 with higher values signifying the asset may be overbought while lower values signify it may be oversold. Pay attention to extreme values for which a trend reversal may happen.
For each data point the trend lines from the previous 50 candles have been constructed and the normalized distance of the current price to the support and resistance line is given. When the price is close to a trend line, a trend reversal can be expected. After several tests on a trendline the price may also breakthrough.Pay attention to price movement when the price is close to either trend line as this is where it is most predictable.
[Instructions]: Predict the detrended data for the next *pred_len* steps given the previous *seq_len* steps.
[Statistics]: The input has a minimum value of *minprice* and a maximum value of *maxprice*, with an average value of *meanval*.

Fig. 7. The extended prompt used in our experiment.

value from 0 to 100. Periods of extreme greed (80–100) or extreme fear (0–20) are often followed by a market high or low, respectively. We collected historical data of the past 6 years using the provided web API.[1]

7. Results for extended forecasting model

Our approach allows the incorporation of specific, yet principled, domain information. Our architecture (Fig. 1) specifically seeks to leverage technical analysis data optimally by using the inverted transformer framework that learns attention over multivariate tokens [17]. In this section, we will study the effect of adding financial information to our model.

7.1. Extended prompt

We found a suitable way to pass additional data to the model is by adding new variables to the dataset over which attention is computed. This is complemented by adding an explanation of the nature of each new variable to the prompt that is fed alongside the data (see Fig. 7).

[1] https://api.alternative.me/

Table 3
Comparison of model performance for 7 day forecast with the addition of technical analysis data. Each subsequent model retains the additional data from previous models.

Dataset	SMAPE	RMSE	Acc. (%)
Basic model (OHLC data, volume)	3.42	1469	52
+ Technical Indicators (Section 6.1.1)	3.80	1449	51
+ SR Trend Lines (Section 6.1.2)	3.31	**1340**	50
+ Market Sentiment (Section 6.1.3)	**3.14**	1360	**54**

This method allows us to activate the LLM's prior knowledge of technical indicators while giving examples of situation where their signal is particularly relevant.

7.2. Effect of adding financial information

The model was originally given the OHLC (Open High Low Close) price data alongside the volume of each candle. The complete dataset was then built progressively with additional information to improve the prediction.

7.2.1. Adding technical indicators

Table 3 indicates that incorporating basic technical indicators such as EMA, volatility, RSI, and ADX yields a modest reduction in RMSE, yet results in an increased SMAPE. The rise in SMAPE can be attributed to its sensitivity to under-forecasting, penalizing predictions that fall below observed values more heavily. These findings suggest that while technical indicators provide some additional information to the model, their contribution is marginal and is offset by a higher bias, as reflected in a decrease in accuracy.

7.2.2. Adding support and resistance trend lines

Including support and resistance (SR) trend lines in the feature set results in a marked enhancement of model precision, as demonstrated by a 9 % reduction in RMSE (Table 3). However, the decrease in SMAPE is comparatively modest at approximately 3 %, and overall forecasting accuracy experiences a slight decline. This outcome mirrors previous patterns observed with technical indicators, where the model tends toward under-forecasting, thereby limiting gains in SMAPE. These results indicate that while additional technical features may improve forecast precision, they simultaneously contribute to model bias. Consequently, sentiment analysis features are incorporated to address this issue and improve the model's overall accuracy.

7.2.3. Adding market sentiment

Introducing the 'fear and greed' index leads to improvements in both SMAPE and overall accuracy, with the RMSE remaining stable (Table 3). The inclusion of market sentiment features appears to counteract the model's prior inclination toward under-forecasting, enhancing balanced predictive performance. This result suggests that supplementing the model with sentiment indicators is an effective means of reducing bias.

Overall, our additions to the dataset were able to improve model performance by around 7 %. We believe this is a promising sign of the ability of the model to learn from new variables, and could be further improved by using more sophisticated technical analysis and sentiment analysis.

7.3. Backtesting of our extended model

To further evaluate the performance of our extended model and its ability to detect signals suitable for trading, we conducted backtesting on out-of-sample data utilizing a straightforward trading strategy based on the model's forecasts. The strategy involves taking a full long position in BTC when the 7-day forecasted return exceeds a threshold of 1 %,

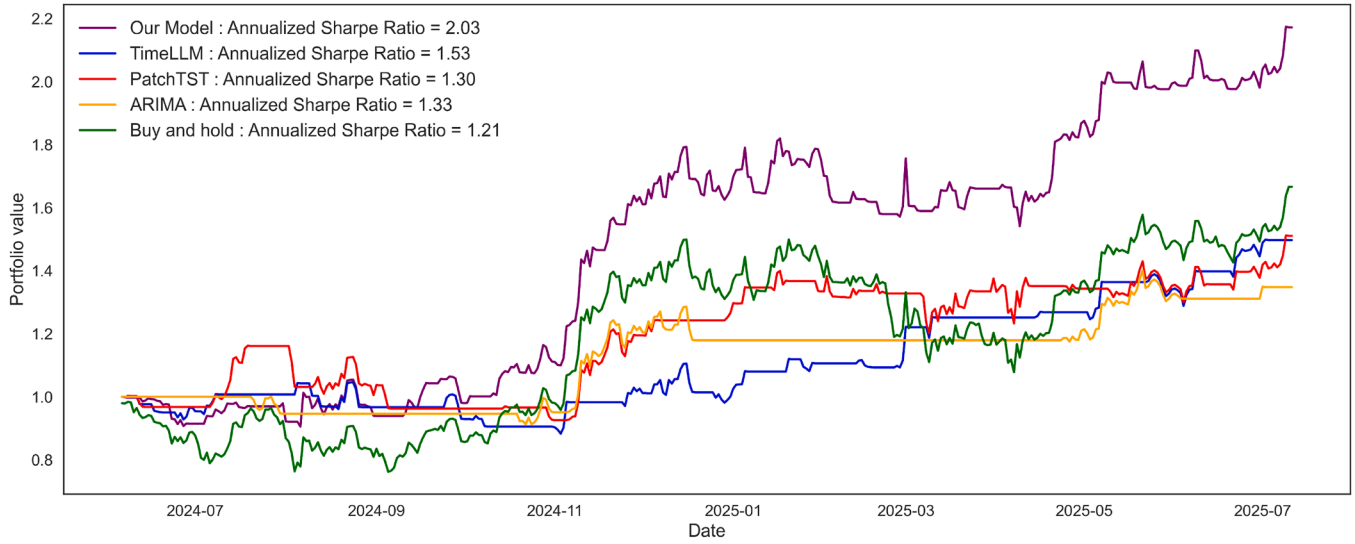


Fig. 8. Backtesting of a simple trading strategy, based on different forecasting models.

Table 4
Summary of a simple trading strategy's backtesting performance, based on different models. A trade is defined as opening a long position in BTC and subsequently selling. Average time per trade refers to the holding period in days before the BTC position is closed. Average gain and loss represent, respectively, the mean profit realized from winning trades and the mean loss incurred from losing trades.

	Our Model	ARIMA	PatchTST	Time-LLM
n° trades	33	6	23	23
Avg time per trade	9 days	18 days	9 days	6 days
Successful trades	58 %	66 %	56 %	60 %
Avg gain	8 %	11.5 %	6.6 %	3.2 %
Avg loss	3.1 %	3.2 %	3.4 %	2.7 %
Sharpe Ratio	2.03	1.33	1.30	1.51
Final portfolio value	2.2	1.34	1.51	1.06

thereby leveraging the model's signal for directional moves consistent with swing trading. The position is held until the forecast falls below 1 %. The 1 % threshold effectively limits over-trading on weak forecast signals. Transaction costs are accounted for with a fixed fee of 0.1 % on each buy and sell transaction, in line with Binance's standard rates. Despite its simplicity, this method yields promising results: the strategy reached a final portfolio value of 2.2 and an annualized Sharpe ratio of 2.0 (in that given year), significantly outperforming the buy-and-hold benchmark, which results in a final portfolio value of 1.6 and a Sharpe ratio of 1.21 (see Fig. 8).

To further assess the capabilities of our model, we implemented the trading strategy using several forecasting models for comparison. ARIMA was included as an industry-standard benchmark, while PatchTST and Time-LLM represent the top-performing architectures from our forecasting benchmarks. Table 4 summarizes the results across these approaches. Our extended model achieves the highest Sharpe ratio and final portfolio value, as well as the greatest number of trades, all while maintaining a high success rate and the second highest average gain per winning trade. In contrast, the ARIMA-based strategy, although conservative with the fewest trades and modest gains, achieved a slightly higher success rate but failed to match the overall profitability of our approach. Importantly, our model was the only one to outperform the buy-and-hold baseline (see Fig. 8), demonstrating that our integration of fractional differencing with prompt-enhanced variate attention enables the LLM to effectively capture trends and signals relevant to trading.

8. Conclusions and future work

In this work, we implemented state-of-the-art methods for time series forecasting on a complex financial dataset, specifically the Bitcoin price chart. Our findings demonstrate that reprogrammed large language models (LLMs) function as efficient forecasters, proving effective even on intricate time series lacking simple seasonality and characterized by substantial stochastic noise.

Building on the Time-LLM model [12], we introduced architectural modifications that resulted in enhanced performance on the Bitcoin dataset, both in terms of average error and when evaluated in a trading strategy. In particular, we examined several detrending techniques and found that the widely used RevIN is suboptimal for financial time series due to significant short-term price fluctuations. Instead, applying fractional differencing enables us to detrend the data while preserving long-term dependencies, which proves especially advantageous during periods of elevated volatility by allowing the model to anticipate larger price movements.

Furthermore, our approach leverages prompting to incorporate additional variables alongside the primary time series, such as technical analysis indicators, leading to further improvements in forecasting accuracy. Despite recent findings by [36] indicating that patching and attention-based embeddings explain much of the observed forecasting performance in LLMs–and acknowledging the extended prediction horizons in their work-our results suggest that augmenting prompts with covariates and external knowledge conveys distinct advantages. Our architecture thus allows targeted integration of domain-specific information within the model's input space. Future research could involve a more comprehensive exploration of financial indicators, potentially integrating other effective variables as well as advanced implementations of automated trend lines. Incorporating sentiment analysis, particularly market sentiment, into prompts may further enhance predictive outcomes. Additionally, evaluating alternative LLM architectures could yield to further improvements.

In summary, our case study highlights the unique benefits brought by the reprogrammed LLM framework. By unifying transfer learning with prompt-based approaches, we facilitate the incorporation of domain-specific knowledge while remaining compatible with patch and embedding models. This synergy results in superior performance for downstream forecasting tasks, and LLMs maintain a distinct advantage in this respect despite recent advances in large-scale time series foundation models [26,27].

CRediT authorship contribution statement

Owen Chaffard: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Pablo Mollá:** Data curation, Investigation, Visualization; **Marc Cavazza:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Conceptualization; **Helmut Prendinger:** Writing – review & editing, Visualization, Supervision, Resources, Project administration, Conceptualization.

Data availability

Data will be made available on request. Reproducible code will be made available on quantlet.com.

Declaration of competing 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.

Acknowledgments

This work was in part supported by the National Institute of Informatics, Tokyo. The first 2 co-authors conducted this work while enrolled in the NII International Internship program. This project was supported by funding from the Horizon Europe research and innovation programme under the Marie Sklodowska-Curie Grant Agreement, acronym: DIGITAL, No. 101119635. The authors would also like to acknowledge the support of the project "IDA Institute of Digital Assets", CF166/15.11.2022, contract number CN760046/23.05.2023; of which the first co-author is a team member.

References

[1] M. Jin, Q. Wen, Y. Liang, C. Zhang, S. Xue, X. Wang, J. Zhang, Y. Wang, H. Chen, X. Li, S. Pan, V.S. Tseng, Y. Zheng, L. Chen, H. Xiong, Large Models for Time Series and Spatio-Temporal Data: A Survey and Outlook, arXiv preprint arXiv:2310.10196 (2023).

[2] S. Bai, J.Z. Kolter, V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, 2018. arXiv preprint arXiv:1803.01271

[3] J. Cao, Z. Li, J. Li, Financial time series forecasting model based on CEEMDAN and LSTM, Phys. A Stat. Mechan. Appl. 519 (2019) 127–139. https://doi.org/10.1016/j.physa.2018.11.061

[4] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Spectral temporal graph neural network for multivariate time-series forecasting, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 6000–6010.

[6] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'23/IAAI'23/EAAI'23, AAAI Press, 2023. https://doi.org/10.1609/aaai.v37i9.26317

[7] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C.C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P.S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E.M. Smith, R. Subramanian, X.E. Tan, B. Tang, R. Taylor, A. Williams, J.X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023. arXiv preprint arXiv:2307.09288

[8] OpenAI, J. Achiam, GPT-4 Technical Report, 2023. arXiv preprint arXiv:2303.08774

[9] T. Zhou, P. Niu, X. Wang, L. Sun, R. Jin, One fits all: power general time series analysis by pretrained LM, in: NeurIPS, 2023.

[10] D. Cao, F. Jia, S.O. Arik, T. Pfister, Y. Zheng, W. Ye, Y. Liu, TEMPO: prompt-based generative pre-trained transformer for time series forecasting, in: The Twelfth International Conference on Learning Representations, 2024.

[11] C. Sun, H. Li, Y. Li, S. Hong, TEST: Text Prototype Aligned Embedding to Activate LLM's Ability for Time Series, 2024, arXiv preprint arXiv:2308.08241

[12] M. Jin, S. Wang, L. Ma, Z. Chu, J.Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, Q. Wen, Time-LLM: time series forecasting by reprogramming large language models, in: The Twelfth International Conference on Learning Representations, 2024.

[13] S. Shanaev, B. Ghimire, A generalised seasonality test and applications for cryptocurrency and stock market seasonality, Quart. Rev. Econ. Finance 86 (2022). https://doi.org/10.1016/j.qref.2022.07.002

[14] S. Singh, A. Pise, B. Moon, Prediction of bitcoin stock price using feature subset optimization, Heliyon 10 5–6 (2024).

[15] M.L. de Prado, Advances in Financial Machine Learning, Wiley, 2018.

[16] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, J. Choo, Reversible instance normalization for accurate time-series forecasting against distribution shift, in: International Conference on Learning Representations, 2021.

[17] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, iTransformer: Inverted Transformers Are Effective for Time Series Forecasting, (2023), arXiv preprint arXiv:2310.06625.

[18] Y. Nie, H.N. Nam, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: long-term forecasting with transformers, in: International Conference on Learning Representations, 2023.

[19] J.J. Murphy, Technical Analysis of the Financial Markets, A Comprehensive Guide to Trading Methods and Applications, Penguin Group, 1999.

[20] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, in: The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference, 35, AAAI Press, 2021, pp. 11106–11115.

[21] H. Wu, J. Xu, J. Wang, M. Long, AutoFormer: decomposition transformers with autocorrelation for long-term series forecasting, in: Advances in Neural Information Processing Systems, 2021.

[22] N. Kitaev, L. Kaiser, A. Levskaya, REFormer: the efficient transformer, in: International Conference on Learning Representations, 2020.

[23] T.M. Duc, M. Cavazza, H. Prendinger, Transformer models for bitcoin price prediction, in: 3rd Int'l Conf. on Frontiers of Artificial Intelligence and Machine Learning (FAIML 2024), 2024.

[24] H. Xue, F.D. Salim, Promptcast: a new prompt-based learning paradigm for time series forecasting, IEEE Trans. Knowl. Data Eng. (2023) 1–14. https://doi.org/10.1109/TKDE.2023.3342137

[25] S.Q. Nate Gruver, Marc Finzi, A.G. Wilson, Large language models are zero shot time series forecasters, in: Advances in Neural Information Processing Systems, 2023.

[26] A.F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S.S. Rangapuram, S. Pineda Arango, S. Kapoor, J. Zschiegner, D.C. Maddix, H. Wang, M.W. Mahoney, K. Torkkola, A. Gordon Wilson, M. Bohlke-Schneider, Y. Wang, Chronos: Learning the Language of Time Series,(2024). arXiv preprint arXiv:2403.07815

[27] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, D. Sahoo, Unified Training of Universal Time Series Forecasting Transformers, (2024). arXiv preprint arXiv:2402.02592

[28] X. Yu, Z. Chen, Y. Ling, S. Dong, Z. Liu, Y. Lu, Temporal Data Meets LLM–Explainable Financial Time Series Forecasting, 2023, arXiv preprint arXiv:2306.11025

[29] E. Plaut, From Principal Subspaces to Principal Components with Linear Autoencoders, 2018, arXiv preprint arXiv:1804.10253

[30] R.B. Cleveland, W.S. Cleveland, J.E. McRae, I. Terpenning, STL: a seasonal-trend decomposition procedure based on loess, J. Off. Stat. 6 (1) (1990) 3–73.

[31] A. Assaf, K. Mokni, I. Yousaf, A. Bhandari, Long memory in the high frequency cryptocurrency markets using fractal connectivity analysis: the impact of COVID-19, Res. Intern. Bus. Finance 64 (2023) 101821. https://doi.org/10.1016/j.ribaf.2022.101821

[32] C.W.J. Granger, R. Joyeux, An introduction to long-memory time series models and fractional differencing, J. Time Series Anal. 1 (1) (1980) 15–29. https://doi.org/10.1111/j.1467-9892.1980.tb00297.x

[33] M.L. de Prado, Advances in Financial Machine Learning, Wiley, 2018.

[34] R. Walasek, J. Gajda, Fractional differentiation and its use in machine learning, Int. J. Adv. Eng. Sci. Appl. Math. 13 (2) (2021) 270–277. https://doi.org/10.1007/s12572-021-00299-5

[35] D. Dickey, W. Fuller, Distribution of the estimators for autoregressive time series with a unit root, JASA J Am Stat. Assoc. 74 (1979). https://doi.org/10.2307/2286348

[36] M. Tan, M. Merrill, V. Gupta, T. Althoff, T. Hartvigsen, Are language models actually useful for time series forecasting?, Adv. Neural Inf. Process. Syst. 37 (2024) 60162–60191.

Owen Chaffard received his master's degrees in theoretical and applied physics from the Ecole Normale Supérieure de Lyon, France. He is currently pusruing a PhD for the MSCA

funded program Digital, a program for European research on digital finance. In particular his work focuses on Grouped Time Series Forecasting in Private Debt Markets and is jointly supervised by CardoAI and RPTU Kaiser-Slautern.

Pablo Mollá is pursuing a master's degree in data science and artificial intelligence at Université Paris Saclay. His work at the National Institute of Informatics focuses on short term forecasting of the Bitcoin price.

Marc Cavazza is a Professor of Artificial Intelligence at the University of Stirling (UK) and a Visiting Professor at NII. He has co-authored over 300 publications in several areas of Applied Artificial Intelligence, including intelligent interfaces and agents. He has been involved in several EU-funded international research projects on these topics. He has also worked in applied neuroscience, particularly neural time series analysis. He has served on the program committee of over 200 conferences and has recently been a reviewer for NeurIPS, ACM Multimedia, ICASSP, ICML, and ICLR. He holds a PhD and an MD from the University of Paris Cité.

Helmut Prendinger received the master's and Ph.D. degrees in logic and artificial intelligence from the University of Salzburg, Austria. He held positions as a research associate and a JSPS post-doctoral fellow with The University of Tokyo. In 1996, he was a Junior Specialist with the University of California at Irvine, Irvine. He is currently a Full Professor with the National Institute of Informatics, Tokyo. His research interests include intelligent drone systems and machine learning (ML), especially deep learning and foundation models, which he applies to the prediction of stock and cryptocurrency price action. He has published more than 260 refereed papers in international journals and conferences. His H-index is 50.