Complex & Intelligent Systems
https://doi.org/10.1007/s40747-024-01400-8

ORIGINAL ARTICLE

Towards efficient similarity embedded temporal Transformers via extended timeframe analysis

Kenniy Olorunnimbe[1] · **Herna Viktor**[1]

Received: 21 February 2023 / Accepted: 22 February 2024

Abstract
Price prediction remains a crucial aspect of financial market research as it forms the basis for various trading strategies and portfolio management techniques. However, traditional models such as ARIMA are not effective for multi-horizon forecasting, and current deep learning approaches do not take into account the conditional heteroscedasticity of financial market time series. In this work, we introduce the similarity embedded temporal Transformer (SeTT) algorithms, which extend the state-of-the-art temporal Transformer architecture. These algorithms utilise historical trends in financial time series, as well as statistical principles, to enhance forecasting performance. We conducted a thorough analysis of various hyperparameters including learning rate, local window size, and the choice of similarity function in this extension of the study in a bid to get optimal model performance. We also experimented over an extended timeframe, which allowed us to more accurately assess the performance of the models in different market conditions and across different lengths of time. Overall, our results show that SeTT provides improved performance for financial market prediction, as it outperforms both classical financial models and state-of-the-art deep learning methods, across volatile and non-volatile extrapolation periods, with varying effects of historical volatility on the extrapolation. Despite the availability of a substantial amount of data spanning up to 13 years, optimal results were primarily attained through a historical window of 1–3 years for the extrapolation period under examination.

Keywords Deep learning · Financial price prediction · Temporal Transformer · Stock market forecast · Multi-horizon forecast · Hyperparameter optimisation

Introduction

The use of historical time series data to make predictions about the prices of financial instruments such as stocks, bonds, and futures is an important area of research and represents an integral component of automated trading and portfolio management systems [1, 2]. In recent years, deep learning approaches have outperformed classical machine learning methods in a variety of tasks, including financial applications. Transformer-based models have emerged as state-of-the-art in deep learning approaches. The development of Transformer-based large language models (LLM) such as BERT [3] and InstructGPT [4] has propelled the advancement of artificial intelligence and its prevalence in public discourse. Transformer models are increasingly applied in the financial domain, and they have demonstrated the potential to outperform traditional statistical methods [5].

The Transformer is state-of-the-art in sequential machine learning [6]. It simplifies learning complexity by eschewing the ordered input dependencies using positional encoding, making learning more efficient and effective, especially for longer sequences [7]. The architecture consists of a stack of encoder and decoder layers that learn the relationship between input and output sequences using multi-head attention, each layer consisting of multiple *attention mechanisms* and a fully connected feed-forward network [8].

An attention mechanism consists of learnable weighted combinations of the elements in two sequences. Each position in one of the sequences connects to every position in the other sequence, capturing long-range dependencies. The Transformer architecture uses attention mechanisms both between and within the encoder and decoder sequences. The

Kenniy Olorunnimbe and Herna Viktor contributed equally to this work.

✉ Kenniy Olorunnimbe
kenniy.olorunnimbe@uottawa.ca

✉ Herna Viktor
hviktor@uottawa.ca

[1] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada

Published online: 08 April 2024

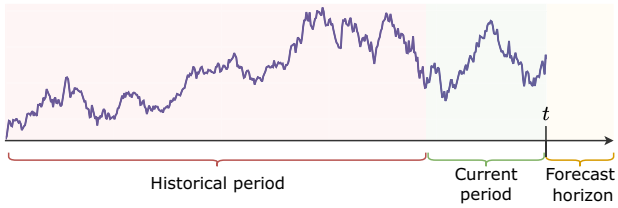


Fig. 1 A financial time series with historical price changes, highlighting the current period and forecast horizon

latter mechanisms are called self-attention or intra-attention [8]. Computationally, the self-attention mechanism is based on key, query, and value vectors, generated from different parts of the input sequence. These vectors are used to compute a weighted sum of the value vector. The weights are determined by the matrix dot product of the query and key vectors, scaled by the inverse square root of the dimension of the key vector. This weighted sum is the output of the self-attention mechanism.

In a time series scenario, the key vectors are generated from the historical data (e.g. the last 2 years) and they represent the data that the model is looking at. The query vectors are generated from the input data (e.g. the last 30 days), representing the questions the model is asking about the historical data. The value vectors represent the answers to those questions (i.e. the prediction horizon). The attention mechanism compares the query vectors to the key vectors and determines how closely they match. The closer the match, the more attention the model pays to that particular data point. The final output of the attention mechanism is a weighted combination of the value vectors, based on the most relevant historical data points. Multi-head attention allows the decoder to search for the most relevant part of the encoder sequence based on the weights learned in the training process.

However, the Transformer model is not sensitive to *local context* in time series data and assumes homoscedasticity, or constant mean and variance, in the attention search space [8]. Local context refers to the tokens or feature vectors immediately surrounding a given input sequence from the historical data sequence. The traditional dot product self-attention mechanism used in the original representation of the Transformer model does not take into account the local context of the input vector, leading to anomalies and optimisation issues [9]. This may also make the model less effective in capturing the complex relationships that exist in financial time series data, which are often not independent and identically distributed (IID) and have non-constant variance.

We introduce novel similarity embedded temporal Transformer (SeTT) architectures, which address the issue of heteroscedasticity in financial data by incorporating similarity vectors into the temporal Transformer [10]. The SeTT and run-similarity embedded temporal Transformer (r-SeTT) models employ proven statistical techniques to address the non-constant variance of financial data and enhance the performance of a baseline temporal Transformer model. We conduct extensive experimentation to assess the efficacy of the models via hyperparameter optimisation and extended timeframe analysis. The results of these experiments demonstrate the improved performance of our algorithm compared to other state-of-the-art models for time series data, including classical financial approaches. The key contributions of this work are:

- We introduce novel similarity embedded temporal Transformer (SeTT) architectures that incorporate historical similarity trends, based the intrinsic properties of financial data and proven statistical principles, into the Transformer model for financial time series forecasting.
- The study involves an extensive evaluation process including over 15,000 optimisation trials across multiple years, timeframes, and market conditions, therefore providing a robust and reliable tool for financial analysis.
- Our analysis indicates that for the same extrapolation period, optimal results were achieved with shorter historical windows of 1–3 years. This is a valuable insight for financial forecasting.
- Our results demonstrate improved empirical performance in multi-horizon forecasts, compared to traditional financial methods and other deep learning methods.

The rest of the paper is organised as follows: in "Related work" section, we review related work in the field of financial time series prediction, focussing on models employing the Transformer model. We describe temporal Transformers and how they are different from the original Transform architecture in "Temporal Transformers" section. This is followed by "Similarity embedded temporal Transformers" section, in which we provide details of temporal similarity embedded Transformer models and our approach to the issue of heteroscedasticity in financial data. The data, experimental setup, and methodology used to evaluate the models are described in "Experimentation" section. In "Results and discussion" section, we present the results of our experiments and discuss their implications. Finally, in "Conclusion" section, we offer concluding remarks and suggest directions for future research.

Related work

The use of deep learning techniques in the stock market has gained significant traction in recent years [10, 11]. Among the various deep learning architectures, Transformers have emerged as state-of-the-art for various sequential learning

tasks, including financial time series analysis [5, 12]. Early works are primarily centred on natural language processing (NLP), focussed on generating feature vectors using available LLM models, and using these features in downstream models such as long short-term memory (LSTM) for the actual forecast [13]. However, more recent studies have begun to use Transformer-based models as the primary model for forecasting. In Ref. [14], the Transformer model is compared with models based on recurrent neural network (RNN) and convolutional neural network (CNN) and is shown to perform favourably. However, the lack of temporal considerations in the study raises questions about the model's effectiveness, especially in comparison with temporal deep learning architectures.

In Ref. [15], a Transformer-based model called Muformer is proposed to improve the predictive accuracy of forecasting problems. It consisted of three main components. First, multiple perceptual domain (MPD) processing mechanisms enhance features by processing input data into multiple outputs in different perceptual domains. The MPD mechanism provides a means of improving the efficiency of Transformer models in long sequence time series forecasting tasks by dividing the input into multiple outputs with different granularity and building local data dependency relationships. Second, a multi-granularity attention head mechanism uses the outputs of the MPD mechanism to reduce the generation of redundant information. Lastly, an attention head pruning mechanism prunes similar or redundant information to enhance model expressiveness. Muformer shows improvement compared to other methods, but it is not clear how practical its long-horizon forecasting is (hourly and in 15-min intervals for 4 months). Additionally, the split between training and testing for validation is quite long, potentially ignoring the temporal dependencies and modelling considerations specific to financial time series data [2].

The Transformer architecture was combined with multiple variants of generalised autoregressive conditional heteroscedasticity (GARCH) and LSTM models in Ref. [16] to create Multi-Transformer architectures for forecasting stock volatility. To avoid variations in the input due to the number of time series used, the positional encoding in their models uses a modified wave function. This is different from the sinusoidal function used in the original Transformer implementation, making it dependent on the lag but consistent across different explanatory variables. The proposed hybrid models, which randomly select subsets of training data and combinations of multiple attention mechanisms to produce the final output, demonstrate improved accuracy in predicting volatility in a well-formulated rolling window training regime compared to other autoregressive models [2]. The results presented in Ref. [16] suggest that their models may be effective in responding to events such as the financial crisis of 2008 and the COVID-19 pandemic. It is unclear, however, how their performance compares to that of other state-of-the-art deep learning approaches. Moreover, the attention mechanism lacks consideration for local data dependencies.

The non-temporal Transformer architecture uses positional encoding and self-attention to draw relationships between data points in a sequence. It remains insensitive to its immediate locality, which may be problematic in time series data. Additionally, the default representation of the attention mechanism assumes homoscedasticity in the attention search space [8]. As the Transformer architecture has evolved to better incorporate temporal dependencies, various approaches have been proposed to address the issue of data locality in the context of time series data, since data points that are closer in time are more likely to be correlated. The lack of locality bias is an important drawback of the Transformer model, affecting its use with temporal data. The long-distance feature sequences are purported to have equal weight as a local feature sequence [9].

Chen et al. [17] used numerical market data in combination with feature vectors generated by passing social media data through an LLM and bidirectional encoder representations from Transformers (BERT) as input into Transformer models called Gated Three-Tower Transformer (GT3). In the GT3 model, temporal data are considered through the use of a shifted window tower encoder (SWTE) with Multi-Temporal Aggregation to address locality. The SWTE extracts and aggregates multiscale temporal features from the original numerical data embeddings by partitioning the embedding matrix into local temporal windows and applying a masked multi-head self-attention operation within each window This captures both local and global temporal information in a unified way. In another work, Lim et al. [5] incorporate *locality enhancements* into a temporal Transformer model called Temporal Fusion Transformer (TFT). Locality is addressed in the temporal fusion decoder by the use of the sequence-to-sequence LSTM layer, which extracts local patterns from the time series data. The encoder takes a sequence of past data points as input, while the decoder processes a sequence of future data points. This accommodates situations in which the number of past and future data points differ, enabling the TFT model to take into account the local context of each data point. The outputs are combined using a multi-headed attention mechanism to weigh the importance of the temporal patterns when making forecasts. An ablation study in the same work highlights the significance of using LSTM for *local processing*. They show that when using the Transformer self-attention alone and not incorporating local processing, the model performance decreased, with an almost 30% increase in loss for financial market data.

Importantly, these works employ the default attention search mechanism by assuming the historical sequence is IID. As explained in "Introduction" section, financial time series are not financial time series are not IID, so the data

distribution will not be identical across the different horizons [18]. Failing to explicitly factor this in raises the risk of oversampling time series that are not relevant to the current timeframe in question during the self-attention lookup [10]. The next section provides a detailed explanation of the temporal Transformer. "Similarity embedded temporal Transformers" section provides an elaborate formulation of our approach and a more thorough explanation of the temporal Transformer is given in the following section.

Temporal Transformers

The domain of deep learning in sequential modelling, particularly in NLP, has traditionally relied on RNN and its different variants, such as LSTM. RNN maintains information across sequences using an internal state that gets updated recursively, acting as a summary of past data points [19]. However, this method suffers from several limitations, such as the vanishing gradient problem and difficulty capturing long-range dependencies in sequences [20]. To address these limitations, Vaswani et al. [8] introduce the Transformer architecture, a novel attention-based model which has since become the standard architecture for sequential deep learning. The Transformer's success lies in its use of self-attention mechanisms, which provide improved parallelism and overcome the limitations of traditional RNNs. The attention mechanism is formulated from the query (Q), key (K), and value (V) matrices comprising q, k, and v vectors, all generated from the input sequence. The q vector represents the information being queried, the k vector represents the information being compared against, and the v vector represents the query result to be generated [21]. The attention weights are calculated by taking the dot product of the Q and K matrices divided by the dimension of the k vectors (d_k), before normalising the resulting scores using softmax. These attention weights are then used to compute weighted values of V that represent the attention mechanism [8]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \tag{1}$$

The q, k, and v vectors are computed by passing the input sequence through a linear transformation to learn weights matrices W_q, W_k, and W_v using a feed-forward neural network (FFNN) for the final output. The same weights, W_q, W_k, and W_v, are shared across all sequence elements, allowing for efficient computation of attention scores and representation of entire sequences in parallel. This also enables the Transformer architecture to process input sequences of varying lengths. In the encoder component of the Transformer architecture, input sequences are associated with the order in which they occur in the data sequence using position encoding made of sinusoidal sine and cosine functions. Sinusoidal functions are used to enable the model to extrapolate to any arbitrary sequence length, beyond the ones it was trained on Ref. [8]. The output sequence is subsequently generated from the decoder, with the encoded representation of the encoder as input.

However, the original Transformer model was designed for non-temporal tasks such as NLP and is not well-suited for handling sequential data with temporal characteristics, such as speech or time series data. Positional encoding is insufficient to capture temporal relationships or local dependencies between elements in a sequence, since it uses a fixed positional representation of each element. Temporal Transformers with locality awareness using learnable positional encoding, such as LSTM or CNN,, were developed to solve this issue [5, 9]. Furthermore, despite the inherent interpretability of the Transformer model stemming from its attention mechanism [8], it cannot distinguish the relative importance of features across different time steps.

Learnable positional encoding enables the encoding of temporal relationships between local data points; this information is used to search through the attention network. It extends the standard multi-head attention mechanism to account for temporal relationships between data points at relatively short proximity. In TFT,, an LSTM sequence-to-sequence layer is used for local processing and to establish short-term relationships within the encoder and decoder, while the self-attention network is used for long-term dependencies [5]. All of the past information within a finite look-back (current) window is incorporated into each sequence element using LSTM.

The value of ξ_t is derived from the original sequence element, X_t, after a non-linear transformation at each time t. The q, k, and v vectors are not derived from ξ_t, but rather from the output of the sequence-to-sequence layer, $\phi(t, n)$, with locality-aware temporal features in a procedure called local processing [5]. $\xi_{t-l:t}$ is fed into the LSTM encoder and $\xi_{t+1:H}$ is fed into the decoder, generating context vectors as a uniform temporal feature set, $\phi(t, n) \in \{\phi(t, -l), \ldots, \phi(t, H)\}$ for each timestep. n is the position index of t in the sequence, l is the lookback window size, and H is the forecast horizon. Local processing captures local temporal patterns and relationships between adjacent time steps via the sequence-to-sequence layer, providing useful local features before longer term dependencies are modelled in the self-attention layer.

This architecture was shown to perform well on various sequential data tasks, including electricity, traffic, and financial volatility forecasts. It captures the interpretability of different time steps and analyses global temporal dynamics using shared weights for the Q, K, and V matrices in each attention head, aggregated across all heads. However, no strong, persistent patterns were observed for the financial

data under experimentation because of the "high degree of randomness". We posit that this is because of the conditional heteroscedasticity of financial time series data.

This study extends the TFT architecture, focussing on financial time series with consideration for financial data characteristics. To assess the efficacy of our models, we conduct extensive experimentation via hyperparameter optimisation and extended timeframe analysis. We aim to optimise their performance to provide a fair comparison between models. This is particularly important when working with complex algorithms and unique data [22]. It allows us to accurately assess the strengths and weaknesses of each model and make informed decisions across different timeframes. In the next section, we provide details of our SeTT architectures [10], which employ the notion of similarity vectors to make the temporal Transformer aware of the distributions of domain-specific time series.

Similarity embedded temporal Transformers

Building on the importance of locality for time series, we explore historical precedence in financial time series evaluation. Specifically, we make use of the lookback window to provide a contextual representation of a deep learning architecture for financial time series data. A financial time series is modelled as finite historical targets $y_{t,t\in\{1,\ldots,T\}}$, where y_t is the target at the current point in time, $y_{\tau,\tau\in\{1,\ldots,H\}}$ is the forecast horizon, and $y_l \leftarrow y_{i,i\in\{T-l+1,\ldots,T\}}$ are the targets of the most recent time series of size l (i.e. the current window). Our SeTT architecture introduces the concept of a sliding window to learn similar time series.

Our similarity embedded temporal Transformer (SeTT) algorithm consists of four steps. (1) A current window and multiple historical windows are generated from the time series data. (2) The historical windows are individually compared with the current window. These comparisons return vectors of $1s$ if similar or $0s$ if dissimilar, each of size l. (3) The individual vectors generated by the comparison are combined into a similarity vector. (4) The similarity vector is embedded into the temporal Transformer architecture during training to mute the dissimilar time series. Algorithm 1 depicts the pseudocode for the SeTT algorithm.

Algorithm 1 Similarity embedded temporal Transformer (SeTT)

$X_{t,t\in\{1,\ldots,T\}}$ ▷ time series features of size T
$y_{t,t\in\{1,\ldots,T\}}$ ▷ time series targets (Δprice) of size T
l ▷ current window size; $l < T$
Ensure: $\hat{y} \sim y_{\tau,\tau\in\{1,\ldots,H\}}$ ▷ vector of H future quantile forecasts

$SimVec \leftarrow$ GENERATESIMVECy, T, l
$Q \leftarrow W_q X, K \leftarrow W_k X, V \leftarrow W_q X$ ▷ initialise attention
$\mathcal{L}(y, W) = L_q(y, \hat{y}, q)$ ▷ update W_q, W_k, W_v by minimising quantile loss
repeat
  A(Q, K, V) $\leftarrow$ softmax $\left(\frac{Q\{K\cdot SimVec\}^T}{\sqrt{d_k}}\right) V \cdot SimVec$
  $\hat{y} \leftarrow L_q(y, \hat{y}, q)$
until stopping condition is met

procedure GENERATESIMVEC(y, T, l)
  $y_l \leftarrow y_{\{T-l+1,\ldots,T\}}$ ▷ current window
  $N \leftarrow T - l$ ▷ size of historical windows
  $\mathbf{A} \leftarrow []$ ▷ initialise an array buffer
  for all j in $0 \ldots N - l - 1$ **do**
    $y_j \leftarrow y_{\{j,\ldots,j+l\}}$ ▷ index of sliding historical windows
    $c \leftarrow$ COMPAREWINy_l, y_j ▷ $c \in \{0, 1\}$
    $\mathbf{a} \leftarrow (\vec{1}_j, \vec{c}_l, \vec{1}_{N-l-j})$ ▷ pad both sides of l-sized vector of repeated c with $1s$
    $\mathbf{A}[j] \leftarrow \mathbf{a}$
  end for
  $\mathbf{h} \leftarrow \min_i\{\mathbf{A} = \mathbf{a}_{ij}\}$ ▷ column-wise minimum values of 2D-array
  return $(\mathbf{h}, \vec{1}_l)$
end procedure

procedure COMPAREWIN(y_l, y_s)
  if $S(y_l, y_s) > \alpha$ **then return** 1 ▷ $\alpha = 0.05$; S: similarity-test function
  else return 0
  end if
end procedure

Historical windows

The financial market is known for its inherent unpredictability and volatility, which make the utilisation of past stock prices and performance a crucial aspect of the analysis process. However, this requires some caution; according to the random walk hypothesis [23], stock price fluctuations follow a random pattern and are independent of one another, making it unfeasible to make predictions solely based on historical movements. Furthermore, two random processes, such as stationary time series (i.e. those with non-changing mean and variance) within different fixed windows, can be considered similar if they exhibit the same statistical properties [24]. This implies that although historical price movements are considered to be random, there may exist specific timeframes which exhibit similar patterns in these movements. We incorporate this concept into a temporal Transformer architecture by generating historical windows from historical price movements of a finite length to identify similar historical patterns.

This forms the foundation of our algorithm to compute the similarity vector. The choice of historical length is a trade-off–extended length may allow the model to identify long-term trends, but it also elevates the probability of capturing irrelevant volatility. As a result, time series models may not be suitable for a very long historical period [25]. On the other hand, while a shorter timeframe may not encompass long-term patterns, it offers a more accurate portrayal of current market conditions. To derive the most advanta-

geous results from both short-term and long-term market data, we propose to compare the representations of historical data within discrete timeframe windows with the most recent short-term data point at the present moment, y_t. The historical windows comprise all the available windows $y_N \leftarrow y_{i,i\in\{1,\ldots,T-l\}}$ of length l that are not in the current window y_l. In the first variant of our SeTT algorithm, these windows are generated by sliding a vector of size l across the historical timeframe of length N using the time series target values, y.

Financial markets often exhibit a high degree of volatility, characterised by significant fluctuations in stock prices. This volatility can result in frequent uninterrupted sequences of either positive or negative price changes, called runs, which have been shown to have a significant impact on the trading landscape [18]. Research suggests that investors consider stocks with shorter up-and-down movements to be less risky than those with longer run-length, leading to further bias in price movement [26]. To consider runs, we introduce the second variant of our SeTT algorithm, called r-SeTT. In r-SeTT, we incorporate the binary representation of the time series targets (Δprice) within the specified time horizon. That is,

$$y_r = \textsc{Run}(y_i) = \begin{cases} 1 & \text{if } y_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where y_i is a price change and y_{ri} is the binary representation of the price change. For the new variant, we replace y in Algorithm 1 with y_r in Equation 2 before comparison. The historical run windows at y_t are all the available positive (1) or negative (0) price change windows of length l, $y_{rN} \leftarrow y_{ri,i\in\{1,\ldots,T-l\}}$, not in the current run window $y_{rl} \leftarrow y_{ri,i\in\{T-l+1,\ldots,T\}}$. We believe that run is an indication of a changing market regime and will be reflected in how historical windows compare with the most recent timeframe.

Comparison between windows

After the historical windows are generated, they are individually compared to the current window. To test for similarity on the SeTT algorithm, we employ Cramér–von Mises (cm), Kolmogorov–Smirnov (ks), and Epps–Singleton (es) tests of fit. These are non-parametric tests for the null hypothesis ($H0$) that two independent samples have the same probability [27–29]. $H0$ in all tests is that two samples are drawn from the same distribution.

Theorem 1 (Two-sample Cramer–von Mises test) *Suppose two independent observations* $X^m \rightarrow \mathbb{R}^m = x_1, x_2, \ldots, x_m$ *and* $X^n \rightarrow \mathbb{R}^n = x_1, x_2, \ldots, x_n$ *with the empirical distribution functions* $F_m(x)$ *and* $G_n(x)$ *and the combined empirical distribution of the entire observation,* $H_{m+n}(x)$*, defined as*

$$F_m(x) = \frac{1}{m}\sum_{i=1}^{m}(X_i \leq x);\ G_n(x) = \frac{1}{n}\sum_{i=1}^{n}(X_i \leq x) \tag{3}$$

$$H_{m+n}(x) = \frac{mF_m(x) + nG_n(x)}{m+n} \tag{4}$$

The Cramer–von Mises statistical test for the null hypothesis that X^m *and* X^n *are independent and identically distributed (IID) is given by* [29]

$$\text{cm test statistic} = \frac{mn}{m+n}\int_{-\infty}^{\infty}\{F_m(x) - G_n(x)\}^2 n H_{m+n}(x) \tag{5}$$

Theorem 2 (Two-sample Kolmogorov–Smirnov test) *Let* X^m *and* X^n *be the same two independent observations with the same empirical distribution function. The Kolmogorov–Smirnov statistical test evaluates the null hypothesis that the two samples are IID is given by* [27]

$$\text{ks test statistic} = \max|F_m(x) - G_n(x)| \tag{6}$$

Theorem 3 (Two-sample Epps–Singleton test) *Let* X^m *and* X^n *be the same two independent observations. The empirical characteristic function of each sample is defined as*

$$\varphi_m(x) = \frac{1}{m}\sum_{i=1}^{m} e^{\sqrt{-1}xX_i};\ \varphi_n(x) = \frac{1}{n}\sum_{i=1}^{n} e^{\sqrt{-1}xX_i} \tag{7}$$

The Epps–Singleton statistical test to evaluate the null hypothesis that X^m *and* X^n *are IID is given by* [29]

$$\text{es test statistic} = \max|\varphi_m(x) - \varphi_n(x)| \tag{8}$$

Corollary 1 (Compare windows for similarity) *Let* l *be the fixed size of the current window,* y_l*. The same size is used for each of the sliding windows,* y_i*, in the historical data. The* p *value is computed based on the values in the distributions to be compared, and the null hypothesis is rejected if the* p *value is less than 5%* ($\alpha = 0.05$)*. Otherwise, it is accepted. A consistent sequence of similar historical windows is determined from the historical windows. Windows similar to* y_l *are represented as an* l*-sized vector of* 1*s* ($\vec{1}_l$)*, whereas dissimilar windows are represented by an* l*-sized vector of* 0*s* ($\vec{0}_l$)*. That is,*

$$\vec{c_{il}} = \begin{cases} \vec{1}_l & \text{if } \textsc{CompareWin}(y_l, y_j) \geq \alpha \\ \vec{0}_l & \text{otherwise} \end{cases} \tag{9}$$

Since the input to the r-SeTT similarity vector routine is a binary representation, we use the *(hd)* to measure the similarity between windows. The Hamming distance captures the

positions of differences between paired binary windows [30], thereby determining the similarity of different run windows:

$$D_{\mathrm{H}}(y_l, y) = \sum_{i=1} |y_{il} - y_i| \tag{10}$$

where y_{il} and y_i are the ith elements of the windows been compared. We used 0.7 as the threshold for the minimum proportion of disagreement, meaning that two windows are considered dissimilar if less than 70% of the paired elements are the same. This is consistent with current knowledge on thresholded Hamming distance search [31].

Similarity vector

A stock symbol is the set of letters representing a publicly traded company or asset on the stock market. For each stock symbol, a similarity vector of length T is generated from the vectors produced by comparing the historical windows with the current window. In Algorithm 1, we defined T as the total size of the time series, l as the current window size, and $N = T - l$ as the size of the historical window.

Corollary 2 (Similarity vector) *Let $M = N - l + 1$. c_i is the binary result of the comparison between the current window and the window at $i \in \{1, \ldots, M\}$. A vector of binary values is first computed from the historical windows as follows:*

$$\mathbf{A}^{MxN} = \{\mathbf{a}_i \in \{0, 1\}^N, i = 1, \ldots, M\} \tag{11}$$

$$= [\mathbf{a}_1, \ldots, \mathbf{a}_M] \tag{12}$$

$$\text{where } \mathbf{a}_i = (\vec{1}_{i-1}, \vec{c}_{il}, \vec{1}_{M-i}) \mid \forall i \in \{1, \ldots, M\} \tag{13}$$

$$\mathbf{h} = \min_i \{\mathbf{A}^{MxN} = \mathbf{a}_{ij}\} \tag{14}$$

The similarity vector is made up of a binary representation of the current window, depicted as a vector of 1s with size l ($\vec{1}_l$) appended to the binary representation of the historical windows, $\mathbf{h}$:

$$\mathbf{s} = (\mathbf{h}, \vec{1}_l) \tag{15}$$

A series of 1s represents a consistent similarity with the current window, potentially across different time horizons; 0s indicate dissimilarity. The current window is also part of the similarity vector, instantiated as a series of 1s. The vectors are then collapsed using column-wise matrix multiplication to give a vector of the same length as T, as illustrated in Fig. 2. A matrix consisting of individual vectors for n stock symbols, $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n]$, is incorporated into the model architecture.

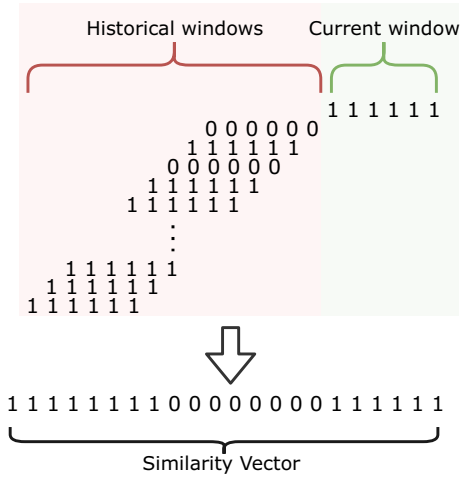


Fig. 2 An illustration of a similarity vector of current and historical windows for a stock symbol

Model architecture

As the system overview in Fig. 3 illustrates, the attention mechanism employs the similarity vector when searching through the temporal attention network. The same similarity vector is used for the entirety of the training period, comprising a constant complexity $O(1)$ addition to the regular temporal Transformer architecture. This ensures that the comparative efficiency of the Transformer model is retained [7]. The Transformer architecture searches the attention mechanism for the most similar encoder sequence in the attention network, assuming that the input data are homoscedastic, with constant mean and variance [8]. In this study, we ensure that during training and inference, the model only searches through historical windows in the attention network that are identical to the current window by muting dissimilar sequences using the similarity vector.

The temporal multi-head attention mechanism captures long-term dependencies based on discrete local processing from the LSTM units. The gated residual network (GRN) is a neural network with gating mechanisms allowing for adaptive depth and complexity. It comprises a residual network (ResNet), giving the model the flexibility of not applying any processing when unneeded (i.e. when the data are small or noisy), adding the benefits of simplicity. By combining the outputs of local processing with the attention mechanisms, the model integrates information about both short-term and long-term dynamics. This provides a robust foundation for financial time series forecasting. For a more in-depth discussion of the temporal multi-head attention mechanism and GRN,, we refer the reader to the work of Li et al. [5], which provides a comprehensive overview of this topic.

Temporal Transformer models are designed to forecast multiple horizons and are trained using past observations

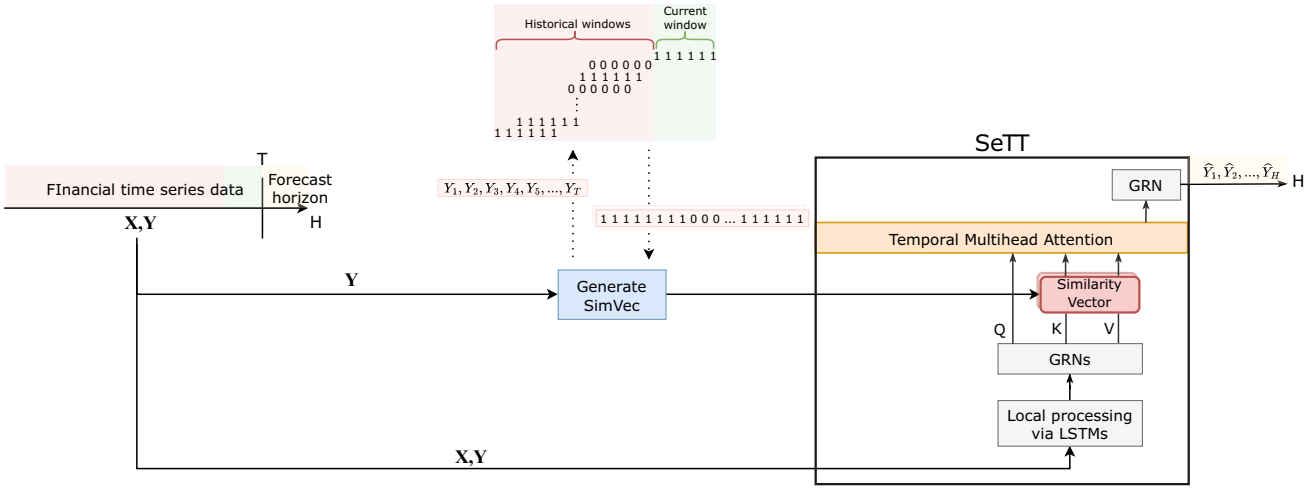


Fig. 3 System overview showing the matrix of similarity vectors for multiple stock symbols generated from the time series data and used within the SeTT architecture. The similarity vectors are used to mute irrelevant keys and values vectors from the queries vectors of the attention mechanism

combined with past and future known patterns (e.g. the day of the week). In contrast to one-step-ahead prediction, *multi-horizon forecast* [32] involves making price prediction multiple steps in advance, as in a trading week (5-day) prediction. This approach is more accurate and efficient than recursive one-step-ahead forecasts when the input involves dynamic and static historical and future attributes, such as financial time series [5, 32].

The future value of a time series is an unknown random variable in a forecast distribution [25]. Instead of estimating the absolute value, we estimate the probability over a range, known as a *quantile forecast* [25]. A quantile forecast is robust when it is difficult to make absolute predictions, as it provides probabilistic forecasts rather than assuming future distribution [32]. This is also useful for minimising risks associated with a financial decision, as it provides the best-case and worst-case scenarios for the prediction target over a probability range [5]. In the training process, predictions are made across all quantiles of interest. The total loss is minimised on multiple forecast horizons using *quantile loss* as the loss function:

$$L_q(y, \hat{y}, q) = \begin{cases} q(y - \hat{y}) & \text{if } y - \hat{y} \geq 0 \\ (1 - q)(\hat{y} - y) & \text{if } y - \hat{y} < 0 \end{cases} \tag{16}$$

$$= \max(q(y - \hat{y}), (1 - q)(\hat{y} - y)) \tag{17}$$

For comparison with other baseline models and consistency with previous work, we used 50% (p50) and 90% (p90) quantiles [5, 9]. The 50% quantile is the target forecast, as it is the median value of the predictive distribution.

Experimentation

Data

We use time series stock market data and financial data to evaluate our model. The Dow Jones Industrial Average (DJIA) is a stock market index weighted by price, consisting of 30 prominent U.S. companies across 20 industries. The stock market symbols for the companies listed in the DJIA are used. We require both market and corporate results data across all timeframes for our experiments. Thus, of the 30 companies available in the index, we only use 20 stocks (Table 1) for which we could obtain the complete data set. They span 15 industries and represent a total weight of 65.62% of the overall index.

Historical market and fundamental data are obtained from SimFin Analytics GmbH (SimFin), an online financial data resource. The time period represented in the data is Jan. 2007 to Nov. 2022. To ensure consistency during the independent trials and aid reproducibility, we used the Lakehouse medallion data design pattern of bronze, silver, and gold tables to refine the input data used in the model [33, 34]. Daily historical US market data and quarterly income data from SimFin are first gathered into bronze tables. Using these bronze tables, we select the relevant fields to create an enriched daily share price silver table for all available market symbols from the source data, using the feature set in Table 2.

As mentioned, we only use 20 of the market symbols represented in the DJIA index. Full historical data are available for these symbols from Nov. 2009 to Nov. 2022, which is a significant portion of the full timeframe available from the source. We create a gold table consisting of just these 20 symbols, enriched with information about their DJIA industry, as that serves as relevant static metadata to the temporal

Table 1 Industry and stock symbols of 20 of the 30 DJIA companies, showing their collective industry weight in the DJIA index

Industry		Symbol	Weight (%)
TECH	Information technology	AAPL, CSCO, IBM, INTC, MSFT, CRM	18.61
HEALTH	Managed health care	UNH	7.88
PHARM	Pharmaceutical, Biopharmaceutical	AMGN, JNJ	6.88
FOOD	Soft Drinks, Food	KO, MCD	5.52
FIN	Financial services	V	4.34
AERO	Aerospace and defense	BA	4.01
CONST	Construction and Mining	CAT	3.73
CONG	Conglomerate	MMM	3.38
APPAREL	Apparel	NKE	2.93
RET	Retailing	WMT	2.69
GOODS	Fast-moving consumer goods	PG	2.61
OIL	Petroleum	CVX	2.07
TELE	Telecommunication	VZ	0.97

Table 2 Features used as input to the model

Attribute	Abbreviation	Description	Type
Open, high, low and close prices	–	Daily open, high, low and close prices	–
Volume	–	Volume of shares traded in a day	–
Shares outstanding	–	All shares available for trade	–
Earnings per share (EPS)	–	Profit for each outstanding share	–
1-day log-price-change	p_logd	Log-change in closing price over 1 day	Target
1-day log-volume-change	v_logd	Log-change in volume over 1 day	Observed numeric
Shares turnover	s_turnover	All shares available for trade	Observed numeric
Price variation	p_variatn	% change in open and close prices	Observed numeric
Price–earning (P/E) ratio	per	Share price relative to EPS	Observed numeric
Normalised high, low, close	norm_h, norm_l, norm_c	High, low and close prices normalised with open price	Observed numeric (x3)
Year, month, weekday	year, month, weekday	Categorical representations of year, month and weekday	Known categorical (x3)
Industry	industry	Market sector of company	Static categorical

Transformer model. The model reads from this table during each experiment, which is filtered in use during the evaluation period, and a monotonically increasing sequence index is added from the start to the end of each period.

Adopting an approach such as the medallion data design pattern for the feature engineering pipeline allows for easier reuse and repurposing of feature data, as specified in the Findability, Accessibility, Interoperability and Reusability (FAIR) principles of scientific data [3]. It also simplifies versioning, facilitating the efficient identification and correction of potential data issues and allowing for updates to be made at any step of the transformation.

Evaluation period

We constructed independent data sets consisting of information about the 20 companies shown in Table 1 across multiple independent extrapolation periods. To determine the periods, we used the volatility index (VIX),[1] a real-time index from the Chicago Board Options Exchange (CBOE). The VIX measures stock market volatility and is based on the relative strength when compared with the S&P 500 index. Note that a VIX of 0–12 is considered low, 13–19 corresponds to normal, and above 20 is deemed high [35]. Higher values are expected during times of financial issues, as during the period of extreme volatility induced by COVID-19 at the start of 2020, as shown in Fig. 4.

We evaluated data within multiple volatile and non-volatile extrapolation periods, with varying amounts of historical (training) data. Figure 5 shows the full list of date ranges used in our experiment. All but the last five trading days within each period are used as training data. After the

[1] investopedia.com/terms/v/vix.asp.

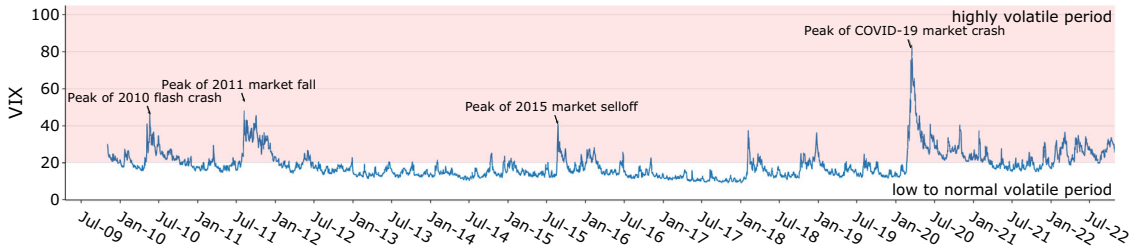
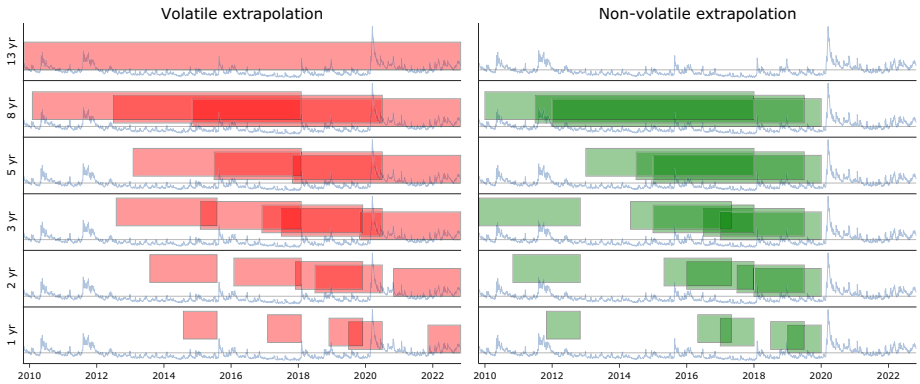


Fig. 4 Volatility index for the period Nov. 2009 to Nov. 2022 (13 years). The lines that fall in the red region are considered to be highly volatile

Fig. 5 Time ranges used in experiments



(a) Time ranges for volatile and non-volatile extrapolation regime using different amounts of training data. Experiments are done on the same extrapolation period with varying training lengths for comparison.

Training length	Volatile extrapolation	Non-volatile extrapolation
13 years	2009-11:2022-10	-
8 years (x3)	2014-11:2022-10 2012-07:2020-06 2010-02:2018-01	2012-01:2019-12 2011-07:2019-06 2010-01:2017-12
5 years (x3)	2017-11:2022-10 2015-07:2020-06 2013-02:2018-01	2015-01:2019-12 2014-07:2019-06 2013-01:2017-12
3 years (x5)	2019-11:2022-10 2017-07:2020-06 2016-12:2019-11 2015-02:2018-01 2012-08:2015-07	2017-01:2019-12 2016-07:2019-06 2015-01:2017-12 2014-05:2017-04 2009-11:2012-10
2 years (x5)	2020-11:2022-10 2018-07:2020-06 2017-12:2019-11 2016-02:2018-01 2013-08:2015-07	2018-01:2019-12 2017-07:2019-06 2016-01:2017-12 2015-05:2017-04 2010-11:2012-10
1 year (x5)	2021-11:2022-10 2019-07:2020-06 2018-12:2019-11 2017-02:2018-01 2014-08:2015-07	2019-01:2019-12 2018-07:2019-06 2017-01:2017-12 2016-05:2017-04 2011-11:2012-10

(b) 43 different date ranges spanning the market dataset, across volatile and non-volatile market periods.

training regime, the model is tested on the last 5 days, which constitute the test set.

Hyperparameters

All of the training was performed using a compute node with NVIDIA Tesla P100 GPU and 32 GB memory provided by Compute Ontario (Graham) and the Digital Research Alliance of Canada.

We previously used preset hyperparameters based on limited preliminary experiments [10]. We have also presented a more extensive hyperparameter search using an open framework called Optuna [36]. In the current work, we initiate the hyperparameter search with the following:

- Learning rate: 0.0001–1 in log-step increments
- Dropout rate: 0–0.9 in 0.1-step increments
- Current window: 16, 30
- Similarity function: es, kv, cm for SeTT; hd for r-SeTT

Using the hyperparameters above, we ran 200 trials across 43 different timeframes to get optimal hyperparameters for the comparison experiments. Optimal hyperparameters for each of the four models—SeTT(es), SeTT(kv)(kv), SeTT(cm), and r-SeTT(hd)—are selected from each of the timeframes, paired with the similarity functions described in "4.2" section. Table 3 shows the listed best parameters and model combinations across the different timeframes, as

Table 3 Hyperparameters after 200 individual trials across both volatile and non-volatile extrapolation periods and 43 different timeframes

yr length	Condition	train_loss	trial_id	Dropout	local_win	lr	Kind	sim_func
1	Non-volatile	0.0637	0	0.3	30	0.082	SeTT	es
1	Non-volatile	0.0763	10	0.6	30	0.836	SeTT	es
1	Non-volatile	0.0886	35	0.1	30	0.186	SeTT	cm
1	Non-volatile	0.0875	1	0.0	16	0.001	SeTT	ks
1	Non-volatile	0.0855	18	0.8	30	0.025	SeTT	ks
2	Non-volatile	0.0497	17	0.4	30	0.004	SeTT	es
2	Non-volatile	0.1054	3	0.9	16	0.016	SeTT	ks
2	Non-volatile	0.0655	22	0.2	30	0.183	r-SeTT	hd
2	Non-volatile	0.0668	4	0.9	30	0.042	r-SeTT	hd
2	Non-volatile	0.0744	4	0.9	30	0.042	r-SeTT	hd
3	Non-volatile	0.0691	2	0.2	30	0.001	r-SeTT	hd
3	Non-volatile	0.0995	10	0.7	30	0.795	r-SeTT	hd
3	Non-volatile	0.0636	14	0.7	30	0.004	SeTT	ks
3	Non-volatile	0.0615	0	0.3	30	0.082	SeTT	es
3	Non-volatile	0.0812	42	0.0	30	0.620	r-SeTT	hd
5	Non-volatile	0.0723	4	0.9	30	0.042	r-SeTT	hd
5	Non-volatile	0.1246	3	0.9	16	0.016	SeTT	ks
5	Non-volatile	0.0559	2	0.2	30	0.001	r-SeTT	hd
8	Non-volatile	0.0693	0	0.3	30	0.082	SeTT	es
8	Non-volatile	0.1337	2	0.2	30	0.001	r-SeTT	hd
8	Non-volatile	0.0592	18	0.8	30	0.004	SeTT	es
1	Volatile	0.0930	6	0.3	30	0.013	SeTT	es
1	Volatile	0.0536	4	0.9	30	0.042	r-SeTT	hd
1	Volatile	0.0702	26	0.6	30	0.007	r-SeTT	hd
1	Volatile	0.0938	37	0.2	30	0.001	SeTT	cm
1	Volatile	0.1084	0	0.3	30	0.082	SeTT	es
2	Volatile	0.0836	0	0.3	30	0.082	SeTT	es
2	Volatile	0.0619	4	0.9	30	0.042	r-SeTT	hd
2	Volatile	0.0662	4	0.9	30	0.042	r-SeTT	hd
2	Volatile	0.1133	1	0.0	16	0.001	SeTT	ks
2	Volatile	0.1022	38	0.2	30	0.001	SeTT	ks
3	Volatile	0.0704	0	0.3	30	0.082	SeTT	es
3	Volatile	0.0689	4	0.9	30	0.042	r-SeTT	hd
3	Volatile	0.0752	4	0.9	30	0.042	r-SeTT	hd
3	Volatile	0.0878	2	0.2	30	0.001	r-SeTT	hd
3	Volatile	0.1294	45	0.8	30	0.051	r-SeTT	hd
5	Volatile	0.0843	32	0.1	30	0.002	r-SeTT	hd
5	Volatile	0.0647	4	0.9	30	0.042	r-SeTT	hd
5	Volatile	0.0951	3	0.9	16	0.016	SeTT	ks
8	Volatile	0.0869	0	0.3	30	0.082	SeTT	es
8	Volatile	0.0794	2	0.2	30	0.001	r-SeTT	hd
8	Volatile	0.0768	2	0.2	30	0.001	r-SeTT	hd
13	Volatile	0.0993	2	0.2	30	0.001	r-SeTT	hd

determined from the trials. Most of the best parameters were observed within the first 40 trials.

We compared our model with autoregressive integrated moving average (ARIMA) and GARCH, two models commonly used for predictions in finance as a result of their ability to capture the temporal structure and conditional variance in financial time series [25]. One approach to using GARCH for prediction is to fit it on the residual of an ARIMA model, where the GARCH mean-output is the volatility estimate and is added to the ARIMA output [37]. Along with these two financial models, we also compared our models with other (DL) models supporting multivariate input attributes.

The DL models are Neural Hierarchical interpolation for Time Series forecasting (NHiTS) [38], probabilistic forecasting with autoregressive recurrent networks (DeepAR) [39], RNN [20], and our baseline TFT [5]. Both NHiTS and DeepAR are state-of-the-art forecasting models, but they use different techniques to make predictions. NHiTS incorporates hierarchical interpolation and multi-rate data sampling for improved accuracy and faster computation. It was shown to achieve an average accuracy improvement of 20% over basic Transformer architectures. DeepAR trains an autoregressive RNN model on multiple time series data for probabilistic forecasting. It, too, demonstrated improved performance over leading models.

We also compared our model with the TFT model to demonstrate areas of improvement in our approach. For completion, we added a sequence-to-sequence RNN model to represent the classical deep learning approach for sequential data. We ran 40 experiments for each model to determine the optimal values for the learning rate and dropout. These hyperparameters can be seen in 8. A total of 15,480 Optuna trials were run to select the optimal hyperparameters for the models, which were determined to be 200×43 for the SeTT algorithms and $40 \times 4 \times 43$ for the other DL algorithms.

Results and discussion

In this section, we present the results of our extended work in hyperparameter optimisation of the SeTT and r-SeTT algorithms across extended and diversified timeframes. We conducted a series of hyperparameter optimisation experiments across various timeframes, as discussed in "Hyperparameters" section. Our extended work aims to validate previous research and create a foundation for further research based on our approach [10]. We provide the detailed results below, followed by a discussion of these results.

Results

Table 4 presents the weighted-summary results. The mean absolute scaled error (MASE) was used to evaluate the performance of the SeTT model, in comparison to other state-of-the-art deep learning models and classical financial models, across 43 different volatile and non-volatile periods, using optimised hyperparameters.

The SeTT model always outperformed classical financial models and other state-of-the-art DL approaches, further highlighting its efficacy for predicting stock trends. The performance is similar with p50 and p90 comparison metrics. Table 5 presents a count of the minimum loss metrics for the different models across the different experiments. The table shows the relative consistency of our model's performance across all timeframes using different evaluation loss functions. Overall, we observed a positive relationship between the error and the length of the historical data window up to 2–3 years, after which it becomes mostly negative.

The key contribution of the SeTT models is that historically similar trends can be incorporated into their design, allowing them to make more accurate predictions in contexts where traditional approaches may be less effective. We conducted extensive optimisation trials for robustness and reliability. The results show that the SeTT model outperformed the baseline TFT model in multiple volatile and non-volatile periods, demonstrating its ability to make more accurate predictions in certain challenging market conditions.

We observed the biggest difference in predictive performance when there were many instances of historical volatility close to the extrapolation period, regardless of the volatility during extrapolation. For example, during the 2019 and 2020 non-volatile extrapolation period, higher instances of performance improvement were observed with the SeTT models compared to the baseline SeTT model. For 2019-07, SeTT demonstrated a reduction in error across all historical lengths, as shown in Table 5. The SeTT models exhibited the lowest errors for the 2020-01 period, meaning that they provided the best predictions even with a relatively shorter historical market length. This result is indicative of the robustness and veracity of our models and their ability to absorb local trends.

While the weighted performance of SeTT and TFT appear to be equivalent during the periods of volatile extrapolation, a significant difference is observed upon closer examination of the performance of individual symbols. This raises the important consideration that even if the aggregated performance improvement is marginal, it can be valuable to examine performance improvements at the individual stock level. As demonstrated in Table 4, there was a slight overall improvement of less than 1% during the volatile period from July 2019 to July 2020. However, further analysis revealed that some individual stock symbols exhibited significant per-

Table 4 MASE loss comparison of weighted-summary of our experiments using optimised hyperparameters across 43 timeframes spanning 13 years

Start	End	yr_len	Condition	Arima	Garch	rnn	Deepar	nhits	sett	tft
2011-11-01	2012-11-01	1	non_volatile	1.592 (+105.2%)	1.599 (+106.1%)	0.986 (+27.1%)	0.951 (+22.6%)	1.571 (+102.6%)	**0.776**	0.806 (+4.0%)
2010-11-01	2012-11-01	2	non_volatile	1.54 (+137.8%)	1.545 (+138.7%)	0.963 (+48.8%)	0.799 (+23.4%)	1.353 (+109.0%)	0.656 (+1.3%)	**0.648**
2009-11-01	2012-11-01	3	non_volatile	1.544 (+158.6%)	1.552 (+160.0%)	0.917 (+53.6%)	0.827 (+38.5%)	1.476 (+147.2%)	0.62 (+3.9%)	**0.597**
2016-05-01	2017-05-01	1	non_volatile	1.651 (+134.1%)	1.644 (+133.2%)	0.844 (+19.7%)	**0.705**	0.91 (+29.0%)	0.818 (+16.1%)	0.796 (+12.9%)
2015-05-01	2017-05-01	2	non_volatile	1.567 (+213.1%)	1.556 (+211.0%)	0.701 (+40.1%)	0.647 (+29.3%)	0.864 (+72.5%)	**0.5**	0.55 (+9.8%)
2014-05-01	2017-05-01	3	non_volatile	1.568 (+200.9%)	1.556 (+198.5%)	0.732 (+40.5%)	0.678 (+30.1%)	0.8 (+53.5%)	0.579 (+11.1%)	**0.521**
2017-01-01	2018-01-01	1	non_volatile	1.455 (+89.8%)	1.459 (+90.4%)	0.846 (+10.4%)	**0.767**	0.802 (+4.6%)	0.798 (+4.2%)	0.903 (+17.8%)
2016-01-01	2018-01-01	2	non_volatile	1.361 (+106.9%)	1.365 (+107.6%)	0.752 (+14.4%)	0.701 (+6.6%)	0.672 (+2.2%)	0.675 (+2.7%)	**0.658**
2015-01-01	2018-01-01	3	non_volatile	1.329 (+142.3%)	1.334 (+143.1%)	0.661 (+20.5%)	0.555 (+1.3%)	0.617 (+12.4%)	0.64 (+16.7%)	**0.548**
2013-01-01	2018-01-01	5	non_volatile	1.342 (+149.2%)	1.344 (+149.6%)	0.664 (+23.4%)	0.642 (+19.2%)	0.628 (+16.7%)	0.564 (+4.7%)	**0.538**
2010-01-01	2018-01-01	8	non_volatile	1.31 (+205.0%)	1.311 (+205.2%)	0.601 (+40.0%)	0.593 (+38.2%)	0.592 (+37.9%)	**0.429**	0.511 (+19.0%)
2018-07-01	2019-07-01	1	non_volatile	1.529 (+173.8%)	1.542 (+176.2%)	0.953 (+70.7%)	0.846 (+51.5%)	0.809 (+44.8%)	**0.558**	0.565 (+1.1%)
2017-07-01	2019-07-01	2	non_volatile	1.57 (+130.1%)	1.581 (+131.8%)	0.978 (+43.4%)	0.924 (+35.5%)	0.832 (+21.9%)	**0.682**	0.695 (+1.9%)
2016-07-01	2019-07-01	3	non_volatile	1.638 (+97.8%)	1.641 (+98.2%)	1.145 (+38.3%)	0.963 (+16.4%)	0.868 (+4.8%)	**0.828**	0.829 (+0.1%)
2014-07-01	2019-07-01	5	non_volatile	1.598 (+105.8%)	1.603 (+106.4%)	1.083 (+39.4%)	0.867 (+11.6%)	0.848 (+9.2%)	**0.776**	0.779 (+0.4%)
2011-07-01	2019-07-01	8	non_volatile	1.596 (+141.6%)	1.6 (+142.1%)	1.072 (+62.2%)	0.987 (+49.4%)	0.799 (+20.9%)	**0.661**	0.721 (+9.2%)
2019-01-01	2020-01-01	1	non_volatile	1.302 (+227.9%)	1.305 (+228.6%)	0.525 (+32.1%)	0.46 (+15.8%)	0.672 (+69.1%)	**0.397**	0.399 (+0.5%)
2018-01-01	2020-01-01	2	non_volatile	1.279 (+336.3%)	1.278 (+336.1%)	0.495 (+68.8%)	0.405 (+38.2%)	0.585 (+99.5%)	**0.293**	0.312 (+6.6%)
2017-01-01	2020-01-01	3	non_volatile	1.319 (+219.9%)	1.325 (+221.3%)	0.585 (+42.0%)	0.485 (+17.8%)	0.719 (+74.3%)	0.423 (+2.5%)	**0.412**
2015-01-01	2020-01-01	5	non_volatile	1.31 (+236.5%)	1.308 (+236.1%)	0.508 (+30.4%)	0.49 (+25.9%)	0.681 (+74.9%)	0.419 (+7.6%)	**0.389**
2012-01-01	2020-01-01	8	non_volatile	1.32 (+211.4%)	1.319 (+211.0%)	0.572 (+35.0%)	0.479 (+13.0%)	0.711 (+67.7%)	0.466 (+10.0%)	**0.424**
2014-08-01	2015-08-01	1	volatile	1.737 (+94.8%)	1.737 (+94.8%)	1.207 (+35.4%)	1.048 (+17.5%)	1.392 (+56.0%)	0.896 (+0.5%)	**0.892**
2013-08-01	2015-08-01	2	volatile	1.779 (+93.2%)	1.78 (+93.3%)	1.303 (+41.5%)	1.141 (+23.9%)	1.488 (+61.6%)	**0.921**	1.013 (+10.1%)
2012-08-01	2015-08-01	3	volatile	1.765 (+87.8%)	1.767 (+88.0%)	1.231 (+30.9%)	1.05 (+11.8%)	1.453 (+54.6%)	0.972 (+3.4%)	**0.94**
2017-02-01	2018-02-01	1	volatile	2.253 (+72.8%)	2.256 (+73.0%)	1.502 (+15.2%)	**1.304**	1.644 (+26.1%)	1.347 (+3.3%)	1.362 (+4.5%)
2016-02-01	2018-02-01	2	volatile	2.118 (+85.8%)	2.121 (+86.0%)	1.362 (+19.5%)	1.207 (+5.9%)	1.446 (+26.8%)	**1.14**	1.164 (+2.1%)
2015-02-01	2018-02-01	3	volatile	1.98 (+134.6%)	1.983 (+135.0%)	1.202 (+42.4%)	1.099 (+30.2%)	1.276 (+51.2%)	0.891 (+5.5%)	**0.844**
2013-02-01	2018-02-01	5	volatile	1.981 (+134.8%)	1.984 (+135.2%)	1.253 (+48.5%)	1.062 (+25.9%)	1.224 (+45.0%)	0.883 (+4.7%)	**0.844**
2010-02-01	2018-02-01	8	volatile	1.911 (+153.8%)	1.915 (+154.3%)	1.201 (+59.5%)	1.007 (+33.8%)	1.158 (+53.8%)	**0.753**	0.761 (+1.1%)
2018-12-01	2019-12-01	1	volatile	1.415 (+246.6%)	1.415 (+246.5%)	0.742 (+81.8%)	0.634 (+55.3%)	0.78 (+90.9%)	0.427 (+4.5%)	**0.408**
2017-12-01	2019-12-01	2	volatile	1.413 (+208.0%)	1.41 (+207.3%)	0.763 (+66.3%)	0.652 (+42.1%)	0.742 (+61.6%)	**0.459**	0.471 (+2.6%)
2016-12-01	2019-12-01	3	volatile	1.477 (+182.5%)	1.477 (+182.4%)	0.708 (+35.4%)	0.6 (+14.7%)	0.772 (+47.6%)	**0.523**	0.526 (+0.7%)
2019-07-01	2020-07-01	1	volatile	1.764 (+542.5%)	1.761 (+541.2%)	0.758 (+176.2%)	0.749 (+172.7%)	1.247 (+354.0%)	**0.275**	0.276 (+0.7%)

Table 4 continued

Start	End	yr_len	Condition	Arima	Garch	rnn	Deepar	nhits	sett	tft
2018-07-01	2020-07-01	2	volatile	1.917 (+ 365.9%)	1.915 (+ 365.4%)	0.972 (+ 136.3%)	0.907 (+ 120.5%)	1.561 (+ 279.3%)	0.417 (+ 1.4%)	**0.411**
2017-07-01	2020-07-01	3	volatile	2.024 (+ 318.9%)	2.024 (+ 318.8%)	1.101 (+ 127.9%)	0.978 (+ 102.3%)	1.661 (+ 243.7%)	**0.483**	0.516 (+ 6.7%)
2015-07-01	2020-07-01	5	volatile	2.197 (+ 223.3%)	2.197 (+ 223.2%)	1.187 (+ 74.7%)	1.187 (+ 74.6%)	1.894 (+ 178.7%)	**0.68**	0.685 (+ 0.7%)
2012-07-01	2020-07-01	8	volatile	2.317 (+ 199.3%)	2.317 (+ 199.3%)	1.345 (+ 73.7%)	1.314 (+ 69.8%)	2.069 (+ 167.3%)	0.815 (+ 5.3%)	**0.774**
2021-11-10	2022-11-01	1	volatile	1.903 (+ 277.3%)	1.899 (+ 276.6%)	0.981 (+ 94.6%)	0.968 (+ 91.8%)	1.072 (+ 112.5%)	**0.504**	0.513 (+ 1.7%)
2020-11-01	2022-11-01	2	volatile	2.015 (+ 230.2%)	2.015 (+ 230.2%)	1.093 (+ 79.0%)	1.051 (+ 72.2%)	1.188 (+ 94.6%)	**0.61**	0.666 (+ 9.0%)
2019-11-01	2022-11-01	3	volatile	1.872 (+ 344.7%)	1.866 (+ 343.2%)	0.964 (+ 129.0%)	0.876 (+ 108.2%)	1.074 (+ 155.2%)	0.436 (+ 3.6%)	**0.421**
2017-11-01	2022-11-01	5	volatile	1.953 (+ 261.1%)	1.943 (+ 259.3%)	1.05 (+ 94.2%)	1.011 (+ 86.9%)	1.232 (+ 127.8%)	0.545 (+ 0.8%)	**0.541**
2014-11-01	2022-11-01	8	volatile	2.077 (+ 212.9%)	2.069 (+ 211.8%)	1.196 (+ 80.2%)	1.158 (+ 74.4%)	1.259 (+ 89.6%)	0.775 (+ 16.7%)	**0.664**
2009-11-01	2022-11-01	13	volatile	2.131 (+ 176.9%)	2.124 (+ 176.0%)	1.264 (+ 64.3%)	1.2 (+ 55.9%)	1.295 (+ 68.2%)	0.782 (+ 1.7%)	**0.77**

The minimum values, signifying the optimal results for each symbol, are emphasized in bold

formance improvements ranging from 5 to 72%, as presented in Table 6.

As previously discussed, the impact of long-distance volatility on predictions appears to be less pronounced when extrapolating during volatile periods. Although the exact cause of this discrepancy remains uncertain, it might be attributed to a preference for a shorter historical timeframe caused by the inherent volatility of the period.

To evaluate the difference in error distribution between the various models, we conducted a Mann–Whitney–Wilcoxon test [40]. The Mann–Whitney–Wilcoxon test, a non-parametric statistical method, allows for the comparison of error distribution across different models without making assumptions about the underlying distributions. The results of this test allowed us to reject the null hypothesis for the classical financial models and the state-of-the-art deep learning methods, indicating that our models have significant predictive performance improvement compared to the traditional models. For comparison between SeTT and TFT models, the periods in which the null hypothesis is rejected correspond to the periods with a significant loss improvement in Table 4.

An example of such comparison can be seen in Table 7 for a highly volatile period between Nov. 2020 and Nov. 2022 with a weighted performance improvement of 9%. Overall, the results of this test provide strong evidence for the improved performance of our models compared to the baseline models.

Discussion

The results presented in "Results" section demonstrate the robustness and generalisability of our core research idea, as our proposed model performs consistently across a wide range of timeframes and market conditions. Although the hyperparameters of the comparison models were optimised to achieve the best results under the test conditions, our model outperformed the others during volatile periods and maintained competitiveness in non-volatile periods.

Using the period and result details in Table 6 as an illustrative example, it is evident that without the similarity vector, our model runs the risk of focussing on an incorrect timeframe during the sample extrapolation regime. As Fig. 6a shows, the cost of that misplaced focus is a loss of predictive performance. However, Fig. 6b, c demonstrates that with the similarity vector, the model focuses on the timeframes that are most similar to the current window and the model makes better extrapolations as a result.

Overall, the findings of this study provide strong evidence for the effectiveness of our approach in predicting stock trends and demonstrate its potential value as a useful tool in this domain. Further research may be needed to explore the

Table 5 Minimum loss error counts for all models across extrapolation periods and market conditions

Period end	Condition	Arima	Garch	rnn	Deepar	nhits	Sett	tft
(a) MASE loss function								
2012-11-01	Non_volatile	0	0	0	0	0	1	2
2017-05-01	Non_volatile	0	0	0	1	0	1	1
2018-01-01	Non_volatile	0	0	0	1	0	1	3
2019-07-01	Non_volatile	0	0	0	0	0	5	0
2020-01-01	Non_volatile	0	0	0	0	0	2	3
2015-08-01	Volatile	0	0	0	0	0	1	2
2018-02-01	Volatile	0	0	0	1	0	2	2
2019-12-01	Volatile	0	0	0	0	0	2	1
2020-07-01	Volatile	0	0	0	0	0	3	2
2022-11-01	Volatile	0	0	0	0	0	2	4
	Sum	0	0	0	3	0	20	20
(b) p50 loss function								
Period end	Condition	Arima	Garch	rnn	Deepar	nhits	Sett	tft
2012-11-01	Non_volatile	0	0	0	0	0	1	2
2017-05-01	Non_volatile	0	0	0	1	0	1	1
2018-01-01	Non_volatile	0	0	0	1	0	1	3
2019-07-01	Non_volatile	0	0	0	0	0	3	2
2020-01-01	Non_volatile	0	0	0	0	0	2	3
2015-08-01	Volatile	0	0	0	0	0	2	1
2018-02-01	Volatile	0	0	0	1	0	2	2
2019-12-01	Volatile	0	0	0	0	0	2	1
2020-07-01	Volatile	0	0	0	0	0	3	2
2022-11-01	Volatile	0	0	0	0	0	2	4
	sum	0	0	0	3	0	19	21

(c) p90 loss function

Arima	Garch	rnn	Deepar	nhits	Sett	tft
0	0	0	0	0	2	1
0	0	0	0	3	0	0
0	0	0	0	0	5	0
0	0	0	1	0	2	2
0	0	0	2	0	0	3
0	0	0	0	0	2	1
0	0	0	0	0	2	3
0	0	0	0	0	0	3
0	0	0	0	0	5	0
0	0	0	0	0	3	3
0	0	0	3	3	21	16

full capabilities of the SeTT architectures and to identify any potential limitations or areas for improvement. However, the results of this study offer a promising foundation for continued research and practical applications in financial time series forecasting.

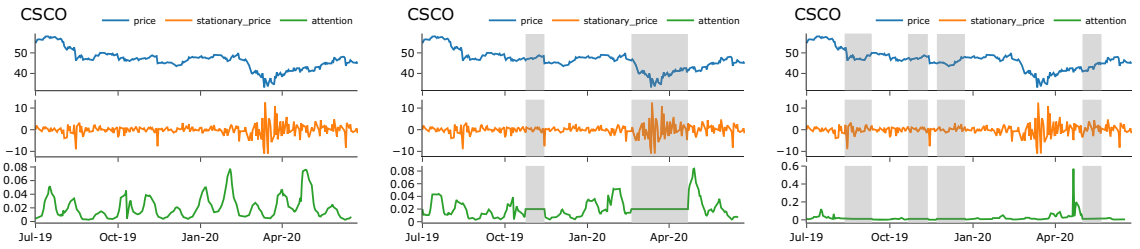
Conclusion

In this study, we demonstrated the extrapolation performance of SeTT models through a comprehensive and rigorous evaluation process focussed on financial forecasts. We conducted over 15,000 optimisation trials across 13 years and 43 different timeframes and used the optimised hyperparameters to compare the SeTT models to other state-of-the-art models for time series data. Using static metadata for 20 symbols from

Table 6 MASE loss comparison of individual stock symbols for the volatile period 2019-07 to 2020-07

Ticker	Arima	Deepar	Garch	nhits	rnn	sett	tft
AAPL	1.617 (+1230.7%)	1.607 (+1222.0%)	0.527 (+333.9%)	0.52 (+327.8%)	0.945 (+678.0%)	**0.122**	0.181 (+49.1%)
AMGN	1.485 (+209.8%)	1.503 (+213.5%)	1.413 (+194.7%)	1.206 (+151.5%)	1.455 (+203.4%)	0.515 (+7.5%)	**0.48**
BA	2.426 (+467.7%)	2.419 (+466.1%)	1.157 (+170.7%)	1.117 (+161.4%)	1.589 (+271.8%)	0.435 (+1.9%)	**0.427**
CAT	1.763 (+612.4%)	1.763 (+612.7%)	0.82 (+231.5%)	0.817 (+230.0%)	1.423 (+475.0%)	0.29 (+17.0%)	**0.247**
CRM	1.683 (+525.8%)	1.679 (+524.3%)	0.971 (+261.2%)	0.806 (+199.8%)	1.075 (+299.9%)	0.315 (+17.2%)	**0.269**
CSCO	1.455 (+696.1%)	1.452 (+694.7%)	0.416 (+127.7%)	0.475 (+159.7%)	0.82 (+348.5%)	**0.183**	0.204 (+11.9%)
CVX	1.832 (+800.3%)	1.834 (+801.2%)	0.446 (+119.4%)	0.485 (+138.3%)	1.21 (+494.8%)	**0.203**	0.249 (+22.5%)
IBM	1.801 (+614.8%)	1.798 (+613.8%)	0.645 (+156.2%)	0.69 (+173.8%)	1.26 (+400.0%)	**0.252**	0.303 (+20.4%)
INTC	1.558 (+1388.3%)	1.568 (+1398.2%)	0.394 (+276.9%)	0.391 (+274.0%)	0.417 (+298.6%)	**0.105**	0.118 (+12.5%)
JNJ	1.629 (+1173.2%)	1.633 (+1176.0%)	0.696 (+444.0%)	0.708 (+452.8%)	1.198 (+836.1%)	**0.128**	0.221 (+72.6%)
KO	1.868 (+571.5%)	1.853 (+566.2%)	0.674 (+142.3%)	0.83 (+198.3%)	1.386 (+398.3%)	0.319 (+14.6%)	**0.278**
MCD	1.632 (+984.9%)	1.635 (+987.3%)	0.351 (+133.4%)	0.467 (+210.7%)	0.78 (+418.4%)	0.225 (+49.6%)	**0.15**
MMM	1.751 (+574.0%)	1.755 (+575.5%)	0.809 (+211.4%)	0.706 (+172.0%)	1.825 (+602.7%)	0.263 (+1.3%)	**0.26**
MSFT	1.66 (+797.8%)	1.654 (+795.0%)	0.575 (+211.3%)	0.568 (+207.5%)	1.142 (+517.8%)	**0.185**	0.194 (+4.9%)
NKE	2.362 (+739.8%)	2.347 (+734.6%)	0.908 (+222.9%)	0.968 (+244.2%)	1.306 (+364.4%)	0.336 (+19.3%)	**0.281**
PG	1.857 (+471.7%)	1.853 (+470.3%)	0.866 (+166.5%)	0.913 (+181.1%)	1.151 (+254.3%)	0.36 (+10.8%)	**0.325**
UNH	1.669 (+554.9%)	1.656 (+549.9%)	0.695 (+172.9%)	0.749 (+194.0%)	1.311 (+414.4%)	**0.255**	0.327 (+28.2%)
V	1.672 (+718.3%)	1.658 (+711.6%)	0.572 (+179.9%)	0.52 (+154.4%)	1.481 (+625.0%)	**0.204**	0.221 (+8.1%)
VZ	2.103 (+495.3%)	2.098 (+494.1%)	0.994 (+181.4%)	1.04 (+194.4%)	1.443 (+308.5%)	0.395 (+11.9%)	**0.353**
WMT	1.449 (+550.4%)	1.45 (+551.0%)	0.524 (+135.4%)	0.579 (+159.7%)	0.541 (+143.0%)	0.234 (+5.1%)	**0.223**

The minimum values, signifying the optimal results for each symbol, are emphasized in bold



(a) Unmuted attention trend (TFT) (b) Attention trend with dissimilar regions (SeTT) (c) Attention trend with dissimilar regions (r-SeTT)

Fig. 6 Attention on the historical window for CSCO (Cisco Systems, Inc.) - 2019-07 to 2020-07

Table 7 Comparison of error distribution using a non-parametric Mann–Whitney test for the volatile period 2020-10 to 2020-10

	mase	p50	p90
SeTT/ARIMA	0.000002	0.000002	0.000002
SeTT/GARCH	0.000002	0.000002	0.000002
SeTT/RNN	0.000002	0.000002	0.000004
SeTT/DeepAR	0.000004	0.000004	0.000105
SeTT/NHiTS	0.000004	0.000004	0.010689
SeTT/TFT	0.023951	0.026642	0.032768

the DJIA index allowed us to evaluate model performance in a range of market conditions and industries. Augmented by the similarity vector, the SeTT model consistently outperforms the others in a range of market conditions and provides improvements over the performance of the TFT in most conditions. The ability to incorporate historically similar trends into the temporal Transformer's architecture contributes to its improved performance with multi-horizon time series forecasts, when traditional approaches may be less effective.

While the TFT model also performs well and competes favourably with the SeTT model, it is unclear which model is superior across various market conditions. This highlights the importance of ongoing research and development in this regard. Rather than choosing a single "winning" model, we suggest that a promising direction for future work is the potential to combine both models in a deep learning ensemble. This approach would take advantage of the strengths of both models and potentially achieve improved performance for multi-horizon financial time series forecasting. Other areas of further research include the use of dynamic metadata such as economic indicators and news events, employing alternative loss functions, and exploring algorithms' applicability to other types of financial data.

In this paper, our model used the same data and feature set as the state-of-the-art models. However, we acknowledge the potential value of techniques such as exploratory data analysis (EDA) and feature importance assessments. In future work, we plan to address the explainability and interpretability aspects of our black box models. We will utilise surrogate models to provide an interpretable approximation of complex Transformer models and improve our understanding of their behaviours [41]. Specifically, we will explore the use of time series multivariate and univariate local explanations (TSMULE) [42], a local surrogate model explanation method specialised for time series that extends the local interpretable model-agnostic explanations (LIME) approach [43]. In addition, we will employ data pre-processing techniques such as EDA,, textual justifications, visualisation, and feature importance assessment to obtain insights during pre-processing and post-processing [44]. Seamlessly integrating these aspects into our framework will not only enhance the models' performance, but will also improve our comprehension of their predictions and the factors that influence model evolution over time.

The current system does not take an online learning approach, and an important direction for future work is the incorporation of near-real-time capabilities. The key properties of online machine learning systems include incremental model updating, adapting to changing data, continuous availability, and cost-efficient inference with low latency [45]. Enabling real-time forecasting would significantly broaden the scope of our work for time-sensitive financial forecasting applications.

Overall, the extensive optimisation trials conducted in this study demonstrated the robustness and reliability of our model, establishing its value as a financial forecasting tool. The results of this study offer a promising foundation for the continued development and use of the similarity vector, and indeed the SeTT model in the field of financial analysis.

Acknowledgements This research was enabled in part by computing resources provided by the Digital Research Alliance of Canada (alliancecan.ca).

Declarations

Conflict of interest On behalf of all the authors, the corresponding author states that there is no conflict of interest.

Appendix: Hyperparameters of comparison between deep learning models

See Table 8.

Table 8 Hyperparameters after 40 individual trials across both volatile and non-volatile extrapolation periods with 4 different models and 43 different timeframes

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
1	Non-volatile	− 1.0548	35	0.0	0.006	deepar	0
1	Non-volatile	0.0804	2	0.0	0.001	tft	16
1	Non-volatile	0.0146	42	0.0	0.034	rnn	0
1	Non-volatile	0.0512	32	0.0	0.001	nhits	0
1	Non-volatile	− 1.2719	44	0.0	0.032	deepar	0
1	Non-volatile	0.0159	39	0.0	0.039	rnn	0
1	Non-volatile	0.1045	1	0.4	0.038	tft	30
1	Non-volatile	0.0615	41	0.0	0.001	nhits	0
1	Non-volatile	0.065	45	0.0	0.0	nhits	0
1	Non-volatile	− 0.9585	21	0.0	0.004	deepar	0
1	Non-volatile	0.0285	42	0.0	0.056	rnn	0
1	Non-volatile	0.0829	11	0.600	0.793	tft	30
1	Non-volatile	− 0.7871	43	0.0	0.045	deepar	0
1	Non-volatile	0.03	39	0.0	0.052	rnn	0
1	Non-volatile	0.0716	3	0.3	0.037	tft	30
1	Non-volatile	0.0807	5	0.0	0.001	nhits	0
1	Non-volatile	0.1091	21	0.0	0.004	nhits	0
1	Non-volatile	0.0271	39	0.0	0.049	rnn	0
1	Non-volatile	− 0.0293	22	0.0	0.01	deepar	0
1	Non-volatile	0.0959	0	0.3	0.082	tft	30
2	Non-volatile	0.0722	4	0.2	0.001	tft	30
2	Non-volatile	0.0176	48	0.0	0.084	rnn	0
2	Non-volatile	− 1.1507	21	0.0	0.004	deepar	0
2	Non-volatile	0.0553	35	0.0	0.0	nhits	0
2	Non-volatile	0.066	3	0.3	0.037	tft	30
2	Non-volatile	− 0.5202	40	0.0	0.013	deepar	0
2	Non-volatile	0.0724	5	0.0	0.001	nhits	0
2	Non-volatile	0.0266	22	0.0	0.015	rnn	0
2	Non-volatile	0.077	13	0.2	0.315	tft	30
2	Non-volatile	− 1.8998	44	0.0	0.018	deepar	0
2	Non-volatile	0.0614	22	0.0	0.0	nhits	0
2	Non-volatile	0.0197	24	0.0	0.027	rnn	0
2	Non-volatile	0.0228	42	0.0	0.029	rnn	0
2	Non-volatile	− 1.2925	37	0.0	0.02	deepar	0
2	Non-volatile	0.0661	1	0.4	0.038	tft	30
2	Non-volatile	0.1005	22	0.0	0.0	nhits	0
2	Non-volatile	− 0.0245	41	0.0	0.011	deepar	0
2	Non-volatile	0.1501	28	0.0	0.002	nhits	0
2	Non-volatile	0.0762	35	0.4	0.009	tft	30

Table 8 continued

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
2	Non-volatile	0.0359	39	0.0	0.049	rnn	0
3	Non-volatile	0.0256	25	0.0	0.018	rnn	0
3	Non-volatile	0.0957	5	0.0	0.001	nhits	0
3	Non-volatile	0.0566	4	0.2	0.001	tft	30
3	Non-volatile	− 0.3923	29	0.0	0.009	deepar	0
3	Non-volatile	0.0161	32	0.0	0.051	rnn	0
3	Non-volatile	− 1.2576	27	0.0	0.014	deepar	0
3	Non-volatile	0.0956	47	0.0	0.0	nhits	0
3	Non-volatile	0.0816	4	0.2	0.001	tft	30
3	Non-volatile	0.0799	43	0.0	0.0	nhits	0
3	Non-volatile	− 0.4596	40	0.0	0.102	deepar	0
3	Non-volatile	0.0162	25	0.0	0.019	rnn	0
3	Non-volatile	0.0745	0	0.3	0.082	tft	30
3	Non-volatile	0.0183	43	0.0	0.056	rnn	0
3	Non-volatile	0.0888	27	0.0	0.0	nhits	0
3	Non-volatile	0.0605	1	0.4	0.038	tft	30
3	Non-volatile	− 0.1673	48	0.1	0.067	deepar	0
3	Non-volatile	0.0331	33	0.0	0.067	rnn	0
3	Non-volatile	0.4716	18	0.1	0.021	deepar	0
3	Non-volatile	0.0799	21	0.2	0.001	tft	30
3	Non-volatile	0.2053	22	0.0	0.002	nhits	0
5	Non-volatile	− 1.0567	27	0.0	0.033	deepar	0
5	Non-volatile	0.1467	27	0.0	0.0	nhits	0
5	Non-volatile	0.0187	41	0.0	0.05	rnn	0
5	Non-volatile	0.0729	10	0.6	0.0	tft	30
5	Non-volatile	0.1068	3	0.3	0.037	tft	30
5	Non-volatile	− 0.2474	36	0.0	0.007	deepar	0
5	Non-volatile	0.0911	23	0.0	0.0	nhits	0
5	Non-volatile	0.0508	26	0.0	0.039	rnn	0
5	Non-volatile	0.068	21	0.2	0.001	tft	30
5	Non-volatile	− 0.8271	34	0.1	0.02	deepar	0
5	Non-volatile	0.0165	44	0.0	0.051	rnn	0
5	Non-volatile	0.0915	27	0.0	0.0	nhits	0
8	Non-volatile	− 1.2994	13	0.0	0.006	deepar	0
8	Non-volatile	0.0215	25	0.0	0.019	rnn	0
8	Non-volatile	0.1639	22	0.0	0.0	nhits	0
8	Non-volatile	0.0593	29	0.3	0.063	tft	30
8	Non-volatile	0.0221	41	0.0	0.082	rnn	0
8	Non-volatile	− 1.3149	25	0.0	0.026	deepar	0
8	Non-volatile	0.1394	41	0.0	0.0	nhits	0
8	Non-volatile	0.1019	43	0.9	0.012	tft	30
8	Non-volatile	− 0.7062	19	0.1	0.029	deepar	0
8	Non-volatile	0.0667	16	0.1	0.943	tft	30
8	Non-volatile	0.1657	44	0.0	0.0	nhits	0
8	Non-volatile	0.0147	42	0.0	0.05	rnn	0
13	Volatile	0.2177	49	0.0	0.001	nhits	0

Table 8 continued

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
13	Volatile	0.0661	32	0.0	0.015	rnn	0
13	Volatile	0.102	1	0.4	0.038	tft	30
13	Volatile	− 0.4961	39	0.0	0.006	deepar	0
1	Volatile	0.076	46	0.0	0.002	nhits	0
1	Volatile	− 0.6419	37	0.0	0.045	deepar	0
1	Volatile	0.0508	39	0.0	0.052	rnn	0
1	Volatile	0.0657	3	0.3	0.037	tft	30
1	Volatile	0.0792	3	0.3	0.037	tft	30
1	Volatile	− 0.8274	27	0.0	0.018	deepar	0
1	Volatile	0.017	30	0.0	0.016	rnn	0
1	Volatile	0.0543	21	0.0	0.004	nhits	0
1	Volatile	0.054	14	0.1	0.005	tft	30
1	Volatile	0.0937	24	0.0	0.002	nhits	0
1	Volatile	0.0182	41	0.0	0.027	rnn	0
1	Volatile	− 1.1668	43	0.0	0.016	deepar	0
1	Volatile	0.0382	35	0.0	0.083	rnn	0
1	Volatile	− 0.6933	42	0.0	0.015	deepar	0
1	Volatile	0.08	32	0.0	0.005	nhits	0
1	Volatile	0.1051	8	0.9	0.042	tft	30
1	Volatile	0.0636	44	0.0	0.001	nhits	0
1	Volatile	0.0381	31	0.0	0.013	rnn	0
1	Volatile	− 0.8287	14	0.0	0.007	deepar	0
1	Volatile	0.1191	3	0.3	0.037	tft	30
2	Volatile	0.1097	23	0.0	0.0	nhits	0
2	Volatile	0.0383	49	0.0	0.034	rnn	0
2	Volatile	0.0827	3	0.3	0.037	tft	30
2	Volatile	− 1.0144	33	0.0	0.056	deepar	0
2	Volatile	0.0165	36	0.0	0.066	rnn	0
2	Volatile	0.0756	24	0.0	0.0	nhits	0
2	Volatile	− 0.6906	32	0.0	0.008	deepar	0
2	Volatile	0.0642	2	0.0	0.001	tft	16
2	Volatile	− 0.6628	33	0.0	0.047	deepar	0
2	Volatile	0.0621	4	0.2	0.001	tft	30
2	Volatile	0.0172	42	0.0	0.052	rnn	0
2	Volatile	0.1239	11	0.0	0.002	nhits	0
2	Volatile	0.1074	27	0.0	0.001	nhits	0
2	Volatile	− 0.1538	42	0.0	0.047	deepar	0
2	Volatile	0.0446	33	0.0	0.051	rnn	0
2	Volatile	0.1103	2	0.0	0.001	tft	16
2	Volatile	0.1217	0	0.3	0.082	tft	30
2	Volatile	− 0.4516	43	0.0	0.048	deepar	0
2	Volatile	0.1313	45	0.0	0.001	nhits	0
2	Volatile	0.0438	44	0.0	0.053	rnn	0
3	Volatile	0.0965	43	0.0	0.0	nhits	0
3	Volatile	0.081	4	0.2	0.001	tft	30
3	Volatile	0.0502	26	0.0	0.015	rnn	0
3	Volatile	− 0.7591	23	0.0	0.015	deepar	0

Table 8 continued

yr length	condition	train_loss	trial_id	dropout	lr	kind	local_win
3	Volatile	− 0.5881	13	0.0	0.006	deepar	0
3	Volatile	0.019	41	0.0	0.026	rnn	0
3	Volatile	0.0856	4	0.2	0.001	tft	30
3	Volatile	0.0797	27	0.0	0.0	nhits	0
3	Volatile	0.1611	24	0.0	0.0	nhits	0
3	Volatile	0.0622	1	0.4	0.038	tft	30
3	Volatile	0.0215	46	0.0	0.023	rnn	0
3	Volatile	− 1.1243	42	0.0	0.01	deepar	0
3	Volatile	0.122	39	0.0	0.001	nhits	0
3	Volatile	0.0285	45	0.0	0.042	rnn	0
3	Volatile	− 0.4661	30	0.0	0.014	deepar	0
3	Volatile	0.0833	17	0.3	0.006	tft	30
3	Volatile	0.0387	44	0.0	0.042	rnn	0
3	Volatile	0.11	0	0.3	0.082	tft	30
3	Volatile	0.1317	42	0.0	0.0	nhits	0
3	Volatile	0.5901	47	0.0	0.013	deepar	0
5	Volatile	0.163	24	0.0	0.002	nhits	0
5	Volatile	0.0463	43	0.0	0.029	rnn	0
5	Volatile	− 0.2628	35	0.0	0.048	deepar	0
5	Volatile	0.0767	0	0.3	0.082	tft	30
5	Volatile	0.0746	31	0.0	0.0	nhits	0
5	Volatile	− 0.2776	29	0.0	0.021	deepar	0
5	Volatile	0.0847	49	0.1	0.014	tft	30
5	Volatile	0.0337	41	0.0	0.067	rnn	0
5	Volatile	− 0.3946	49	0.0	0.083	deepar	0
5	Volatile	0.0798	3	0.3	0.037	tft	30
5	Volatile	0.1749	41	0.0	0.0	nhits	0
5	Volatile	0.0345	46	0.0	0.021	rnn	0
8	Volatile	0.1838	47	0.0	0.0	nhits	0
8	Volatile	0.097	4	0.2	0.001	tft	30
8	Volatile	0.0532	45	0.0	0.091	rnn	0
8	Volatile	− 0.5068	31	0.0	0.045	deepar	0
8	Volatile	0.0298	37	0.0	0.053	rnn	0
8	Volatile	0.102	22	0.0	0.0	nhits	0
8	Volatile	− 0.0927	44	0.1	0.028	deepar	0
8	Volatile	0.0987	24	0.0	0.008	tft	30
8	Volatile	0.0892	34	0.3	0.02	tft	30
8	Volatile	− 0.2856	43	0.0	0.011	deepar	0
8	Volatile	0.1929	41	0.0	0.001	nhits	0
8	Volatile	0.0417	41	0.0	0.099	rnn	0

The loss value for deepar is negative because it uses a negative binomial loss function

References

1. Sezer OB, Gudelek MU, Ozbayoglu AM (2020) Financial time series forecasting with deep learning: a systematic literature review: 2005–2019. Appl Soft Comput 90:106181. https://doi.org/10.1016/j.asoc.2020.106181
2. Olorunnimbe K, Viktor HL (2022) Deep learning in the stock market—a systematic survey of practice, backtesting and applications. Artif Intell Rev. https://doi.org/10.1007/s10462-022-10226-0
3. Wilkinson MD, Dumontier M, Aalbersberg IJ et al (2016) The FAIR guiding principles for scientific data management and stewardship. Sci Data 3(1):1–9. https://doi.org/10.1038/sdata.2016.18
4. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P, Leike J, Lowe R (2022) Training language models to follow instructions with human feedback. https://doi.org/10.48550/arXiv.2203.02155
5. Lim B, Arik SO, Loeff N, Pfister T (2021) Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecast 37(4):1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012
6. de Santana Correia A, Colombini EL (2022) Attention, please! a survey of neural attention models in deep learning. Artif Intell Rev. https://doi.org/10.1007/s10462-022-10148-x
7. Tay Y, Dehghani M, Bahri D, Metzler D (2023) Efficient transformers: a survey. ACM Comput Surv 55(6):1–28. https://doi.org/10.1145/3530811
8. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst. https://doi.org/10.48550/arXiv.1706.03762
9. Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y-X, Yan X (2019) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Proceedings of the 33rd international conference on neural information processing systems. Curran Associates Inc., New York, pp 5243–5253. https://doi.org/10.48550/arXiv.1907.00235
10. Olorunnimbe K, Viktor HL (2022) Similarity embedded temporal transformers: enhancing stock predictions with historically similar trends. In: 26th International symposium on methodologies for intelligent systems (ISMIS), pp 388–398. https://doi.org/10.1007/978-3-031-16564-1_37
11. Soleymani F, Paquet E (2020) Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder-DeepBreath. Expert Syst Appl 156:113456. https://doi.org/10.1016/j.eswa.2020.113456
12. Wen Q, Zhou T, Zhang C, Chen W, Ma Z, Yan J, Sun L (2022) Transformers in time series: a survey. https://doi.org/10.48550/arXiv.2202.07125
13. Mishev K, Gjorgjevikj A, Vodenska I, Chitkushev LT, Trajanov D (2020) Evaluation of sentiment analysis in finance: from lexicons to transformers. IEEE Access 8:131662–131682. https://doi.org/10.1109/access.2020.3009626
14. Wang C, Chen Y, Zhang S, Zhang Q (2022) Stock market index prediction using deep transformer model. Expert Syst Appl 208:118128. https://doi.org/10.1016/j.eswa.2022.118128
15. Zeng P, Hu G, Zhou X, Li S, Liu P, Liu S (2022) Muformer: a long sequence time-series forecasting model based on modified multi-head attention. Knowl-Based Syst 254:109584. https://doi.org/10.1016/j.knosys.2022.109584
16. Ramos-Pérez E, Alonso-González PJ, Núñez-Velázquez JJ (2021) Multi-transformer: a new neural network-based architecture for forecasting s& p volatility. Mathematics 9(15):1794. https://doi.org/10.3390/math9151794
17. Chen J, Chen T, Shen M, Shi Y, Wang D, Zhang X (2022) Gated three-tower transformer for text-driven stock market prediction. Multimed Tools Appl 81(21):30093–30119. https://doi.org/10.1007/s11042-022-11908-1
18. Prado ML (2018) Advances in financial machine learning, 1st edn. Wiley, Hoboken
19. Lim B, Zohren S (2021) Time-series forecasting with deep learning: a survey. Philos Trans R Soc A. https://doi.org/10.1098/rsta.2020.0209
20. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
21. Russell S, Norvig P (2021) Artificial intelligence: a modern approach, global edition, 4th edn. Pearson, Harlow
22. Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316. https://doi.org/10.1016/j.neucom.2020.07.061
23. Malkiel BG (2023) A random walk down Wall street: the time-tested strategy for successful investing, 50th edn. W.W. Norton and Company, New York
24. Brockwell PJ, Davis RA (2016) Introduction to time series and forecasting. Springer texts in statistics. Springer, New York. https://doi.org/10.1007/978-3-319-29854-2
25. Hyndman R, Athanasopoulos G (2021) Forecasting: principles and practice, 3rd edn. OTexts, Melbourne
26. Raghubir P, Das SR (2010) The long and short of it: why are stocks with shorter runs preferred? J Consum Res 36(6):964–982. https://doi.org/10.1086/644762
27. Hollander M, Wolfe DA, Chicken E (2013) Nonparametric statistical methods, 3rd edn. Wiley, Hoboken
28. Goerg SJ, Kaiser J (2009) Nonparametric testing of distributions: the Epps–Singleton two-sample test using the empirical characteristic function. Stata J 9(3):454–465. https://doi.org/10.1177/1536867X0900900307
29. Erlemann R, Lockhart R, Yao R (2022) Cramér–von Mises tests for change points. Scand J Stat 49(2):802–830. https://doi.org/10.1111/sjos.12544
30. Leskovec J, Rajaraman A, Ullman JD (2020) Mining of massive datasets, 3rd edn. Cambridge University Press, Cambridge
31. Ong E-J, Bober M (2016) Improved hamming distance search using variable length hashing. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, NV, pp 2000–2008. https://doi.org/10.1109/cvpr.2016.220
32. Wen R, Torkkola K, Narayanaswamy B, Madeka D (2017) A multi-horizon quantile recurrent forecaster. In: NIPS'17: proceedings of the 31st international conference on neural information processing systems. Curran Associates Inc., New York. https://doi.org/10.48550/arXiv.1711.11053
33. Armbrust M, Ghodsi A, Xin R, Zaharia M (2021) Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In: 11th Conference on innovative data systems research, 8. https://researchr.org/publication/Zaharia0XA21
34. Databricks: what is a Medallion architecture? databricks.com/glossary/medallion-architecture
35. Preston H, Edwards T (2017) A practitioner's guide to reading VIX. S & P Global, New York
36. Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. ACM, Anchorage, pp 2623–2631. https://doi.org/10.1145/3292500.3330701
37. Mustapa FH, Ismail MT (2019) Modelling and forecasting S&P 500 stock prices using hybrid Arima–Garch model. J Phys Conf Ser 1366(1):012130. https://doi.org/10.1088/1742-6596/1366/1/012130
38. Challu C, Olivares KG, Oreshkin BN, Garza F, Mergenthaler-Canseco M, Dubrawski A (2022) N-HiTS: neural hierarchical

interpolation for time series forecasting. https://doi.org/10.48550/arXiv.2201.12886
39. Salinas D, Flunkert V, Gasthaus J, Januschowski T (2020) DeepAR: probabilistic forecasting with autoregressive recurrent networks. Int J Forecast 36(3):1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001
40. Paquet E, Soleymani F (2022) QuantumLeap: hybrid quantum neural network for financial predictions. Expert Syst Appl 195:116583. https://doi.org/10.1016/j.eswa.2022.116583
41. Molnar C (2022) Interpretable machine learning: a guide for making black box models explainable. https://christophm.github.io/interpretable-ml-book/
42. Schlegel U, Vo DL, Keim DA, Seebacher D (2021) TS-MULE: local interpretable model-agnostic explanations for time series forecast models. In: Machine learning and principles and practice of knowledge discovery in databases. Communications in computer and information science. Springer, New York, pp 5–14. https://doi.org/10.1007/978-3-030-93736-2_1
43. Ribeiro MT, Singh S, Guestrin C (2016) "Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. KDD '16. Association for Computing Machinery, New York, pp 1135–1144. https://doi.org/10.1145/2939672.2939778
44. Bai X, Wang X, Liu X, Liu Q, Song J, Sebe N, Kim B (2021) Explainable deep learning for efficient and robust pattern recognition: a survey of recent developments. Pattern Recognit 120:108102. https://doi.org/10.1016/j.patcog.2021.108102
45. Barry M, Bifet A, Billy J-L (2023) StreamAI: dealing with challenges of continual learning systems for serving AI in production. In: 2023 IEEE/ACM 45th international conference on software engineering: software engineering in practice (ICSE-SEIP). IEEE, Melbourne, pp 134–137. https://doi.org/10.1109/icse-seip58684.2023.00017