Highlights

A Meta-learning approach in movement prediction of aperiodic time-series data

Bao Long Nguyen,Kenta Hongo,Ryo Maezono,Bac Le,Tom Ichibha

- Aggregating information from multi-source, multi-period of data improves the model.
- Meta-learning efficiently aggregates information from multi-source, multi-period of time.
- Achieve better results than frequency decomposition method on aperiodic datasets.
- Ablation study shows the superior of Temporal-ML compared to Transformers.

A Meta-learning approach in movement prediction of aperiodic time-series data[⋆]

Bao Long Nguyen[a,**], Kenta Hongo[b], Ryo Maezono[a], Bac Le[c] and Tom Ichibha[a,*]

[a] *School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Nomi, Ishikawa, 923-1292, Japan*
[b] *Research Center for Advanced Computing Infrastructure, JAIST, Nomi, Ishikawa, 923-1292, Japan*
[c] *Deparment of Computer Science, University of Science, Vietnam National University, Ho Chi Minh City, 700000, Vietnam*

ARTICLE INFO

Keywords:
Aperiodic time-series data
Meta-learning
Foreign exchange
`Temporal-ML`

ABSTRACT

Predicting aperiodic time-series data (e.g. stock price, foreign exchange, Bitcoin price, etc.) is a difficult task for machine learning models because eternal factors including politics, economy, and society have a significant impact on the data; The data has a massive and non-stationary variance; Data's strong non-periodicity makes feature extraction challenging. These challenges make the data unstable and non-cyclical. To overcome the above challenges, we proposed `Temporal-ML`, which effectively extracting temporal feature of the data and synthesizing sub-models across multi-source. Experiments on aperiodic data (foreign exchange data of 60 currency pairs over 24 years from 2000 to 2024) show that the proposed method performs well and has 12-14% higher accuracy compared to the `NHITS` - the state of the art (SOTA) model in 2023 on time-series data, in the task of predicting the trend (upward or downward) of the next trading day. Meanwhile, the results obtained on periodic data (Electricity Transformer Temperature, Weather) are approximately `NHITS`. Ablation study is also conducted, which analyze the contribution of each component in the proposed method. Based on the results, this work proves the role of each component as well as the rationality in choosing algorithms in the method designation.

1. Introduction

Time-series data is becoming increasingly diverse not only in terms of data sources (e.g., finance, energy, transportation, etc.), but also in terms of methods for analyzing and forecasting them (e.g., frequency decomposition and methods based on historical data memorization). This makes sense as planning, organizing, and developing business strategies are greatly aided by the analysis and prediction of time-series data.

Two main approaches used in analyzing and predicting time-series data are fundamental analysis and technical analysis [3]. While fundamental analysis concentrates on examining external factors that are hard to capture from past price changes, such as the economic strategies and policies of nations and businesses, technical analysis relies entirely on historical price changes to forecast future trends.

Unfortunately, these methods mainly focus on data with strong periodicity. **Aperiodic time-series data** such as stock prices, foreign exchange rates, has been the subject of very few investigations. The primary characteristic of aperiodic data is that it is **heavily impacted by external influences**, including production strategy and the political and economic policies of nations or businesses. This results in notable modifications to the data's characteristics, which are also referred to as concept drift scenarios [21]. Mathematically, this implies that aperiodic time-series data has a **continuous and unstable variance**. Additionally, because the data no longer exhibits periodicity, it becomes extremely **challenging to extract characteristics** from it using a sliding window.

External influences is the most challenging to resolve using technical analysis techniques as it is difficult to extract political and social information from historical data. However, according to the efficient market hypothesis [10], historical data itself clearly reflects information about market movements. This implies that, by carefully examining the

[⋆] This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

[*] Corresponding author

[**] Principal corresponding author

✉ `s2210434@jaist.ac.jp` (B.L. Nguyen); `hongo@jaist.ac.jp` (K. Hongo); `rmaezono@jaist.ac.jp` (R. Maezono); `lhbac@fit.hcmus.edu.vn` (B. Le); `ichibha@icloud.com` (T. Ichibha)

ORCID(s): 0000-0002-6411-8943 (B.L. Nguyen); 0000-0002-2580-0907 (K. Hongo); 0000-0002-5875-971X (R. Maezono); 0000-0002-4306-6945 (B. Le); 0000-0002-7455-4968 (T. Ichibha)

transaction history, one can completely grasp market fluctuations and forecast the future without aggregating external news. In addition, studies [2, 24] indicate the correlations between the financial indicators of a certain company and the indicators of other companies. In other words, integrating analysis of multiple data sources within the same domain can yield valuable insights for the indicator of interest. However, recent methods concentrate on exploiting insight from single source rather than mining correlations between multiple sources.

Ensemble learning is frequently utilized to mitigate the impacts of variation [1, 31, 37]. This type of learning involves dividing the data into several parts, assuming and learning a certain amount of variation in each part. This approach is reasonable because it is challenging for using only one single model to capture all of the variability in the data over long time. Even so, the synthesis mechanism of ensemble learning methods is still simple and rigid, which may not be able to capture the exploited information in sub-models.

So far, many models have been proposed to be able to extract features on data in general and time-series data in particular. Typically, Convolutional neural network (CNN) [19], Long short-term memory (LSTM) [13], and attention mechanism [4, 23, 35] have been proposed to extract local features, selectively remember long, short-term features, and emphasize vectors in the feature matrix, respectively. However, these methods still have the problem of information forgetting, which limits the ability to learn from long sequences. In addition, recent approaches propose to decompose input signal into frequency bands [7, 22, 41]. These methods are reported to achieve an excellent performance but most of experiments are conducted on highly periodic data. Our data exhibits strong non-periodicity, which makes it a huge challenge for recent methods.

On the other hand, optimization-based meta-learning algorithm (ML) algorithms are known for their ability to increase the generalization and adaptability of a model on limited data [14, 36]. During training, the optimization process of ML can be viewed as an efficient synthesis of models across sub-tasks (sub-datasets). Based on this ability, a machine learning models trained by ML algorithms are expected to efficiently synthesize information from multiple data sources/time periods as well as adapt well to the continuous change of variance.

We approach the above challenges through the combination of temporal feature extraction and model synthesis. Specifically, by using MAML [11], an ML algorithm, we efficiently synthesize the parameters of local models, which significantly reduces variance loss and increases the ability to aggregate information from multiple data sources as well as provides the generalization to the aggregated model. To effectively learn the temporal feature, we use Bidirectional LSTM (BiLSTM) [8], which is an old but efficient extraction method in dealing with sequenced data. Deep learning approaches such as BiLSTM encounter obstacles in learning long sequences because long-term features will be forgotten over time. We overcome this problem by dividing dataset into multiple sequences as approached in ensemble model and synthesize the learned parameters effectively by ML. As a result, these challenges can be handled well.

We demonstrate the superiority of the proposed algorithm by solving the problem of predicting the trend (upward or downward) of time-series data and comparing to NHITS [5] (SOTA model in 2023) on two types of data: (1) - Aperiodic data, which contains USD/JPY exchange rate data (2000-2024) and the exchange rate data of 60 currency pairs made of 18 countries (2014-2024); (2) - Periodic data which contains Electricity Transformer Temperature [40] and Weather [17] dataset. These datasets are publicly available on the Internet and can be downloaded easily. Moreover, we conduct an ablation study to analyze the contribution of MAML and BiLSTM in our method. The official implementation can be found at https://github.com/baolongnguyenmac/multi_fx.

In summary, our main contributions are as follows:

- We propose Temporal-ML, a method that effectively extracts temporal features of aperiodic time-series data and synthesizes sub-models across multi-source data.
- We conduct experiments on aperiodic and periodic data. The result shows that the proposed method outperforms NHITS on aperiodic data and achieves competitive performance on periodic data, which indicates that ML algorithms are efficient in aggregating information from multiple sources.
- Ablation study is conducted to evaluate the contribution of components in proposed method as well as the superior of Temporal-ML compared to Transformers and other models.

2. Related work

2.1. Traditional approach

Two main directions in tackling the problems of time-series data are deep learning methods and ensemble methods. Meanwhile, the former approach tends to extract the deep hidden feature from data; the latter tries to integrate the strength of several sub-models to enhance the predictive ability.

CNN and its variants [19, 32, 39] use a sliding window (kernel) to slide along the time direction of data, which effectively extract the local relationship between the timestamp under consideration and the nearby timestamps. During the training process, CNN tries to learn a suitable kernel to extract the temporal features, making it perform faster compare to conventional neural network. However, for time-series data in general and aperiodic time-series data in particular, the data properties at any given time depend not only on recent times but also on long-term times in the past. Therefore, CNN should only be used as a method to support feature extraction in the process of solving problems on time-series data.

Instead of focusing on extracting local feature, RNN-based methods [18, 30, 33] concatenate on creating a hidden state which selectively presents a long period of time from input data. Indeed, these algorithms perform on an input sequence of arbitrary size. At each step, the hidden state is updated by extracting information from the current input element and combining with the previous state. As a result, RNN-based methods easily outperform CNN since they can effectively capture the historical data. Nevertheless, RNN and its variations face a big problem of forgetting data in tackling long sequences.

Transformers-based methods [7, 41] utilizes the attention mechanism to extract the relationships between each element in the input sequence. Accordingly, we can predict the future value based on its dependence on each individual element in the past. In addition, Transformers can also be used to extract relationships between dimensions in data [21, 38]. Although it has been proven effective on language processing and time-series data prediction tasks, Transformers are only effective if the elements in the input sequence contain a lot of information to extract (high-dimensional vectors). For simple data types such as financial indicators, using Transformers will encounter many difficulties.

On the other hand, aperiodic data is greatly influenced by external factors. These factors are directly reflected in the data of not only the indicator of interest but also other indicators in the same domain. Therefore, exploiting multi-source data can bring valuable information. However, the aforementioned methods have absolutely no mechanism to extract or synthesize this type of information. At this point, ensemble methods (bagging, boosting, stacking models) can be considered an obvious solution since it has a specific mechanism to learn and synthesize information from multiple data sources. However, the simplicity and rigidity in information aggregation of these methods have limited the idea of combining multi-source information. As a result, ensemble models have been gradually forgotten in recent studies.

2.2. Frequency decomposition-based approach

Fourier series has been widely used in the analysis and prediction of time-series data. It was shown by Jean-Baptiste Fourier to be able to approximate any periodic function with sine and cosine basis functions. With this ability, Fourier series can easily identify seasonal patterns such as weekly, monthly seasonality. STL decomposition [29] has similar capabilities but is more powerful because it allows users to change seasonal parameters and handles outlier data well (i.e., outlier data does not affect the trend and seasonal approximation). However, these methods are often inflexible because they depend on fixed parameters and cannot be tuned automatically. As a result, they become less flexible when dealing with complex data such as aperiodic time-series data.

To overcome this drawback, deep learning methods are used to exploit or simulate frequency features. With the flexibility in network architecture and feature extraction capabilities, deep learning methods quickly integrate the frequency features of data into the prediction process. For example, Fractional Fourier Transform [16] uses Fourier transform as a feature extraction step with the goal of transforming the original data to the spectral domain. The features in the spectral domain will then be fed into recurrent networks networks (e.g., GRU, RNN) for further processing.

Beyond incorporating frequency features into the learning process, deep learning networks can also simulate them. For example, NBEATS [27] decomposes the input data into frequency bands using multiple stacked blocks, each of which consists of FullyConnected (with or without non-linear activation function) layers. Each block is trained to predict future values and approximate its input values. The next block learns from the difference between the actual input values and the previous block's input data estimates. The prediction of each block is thus an approximation of a data's frequency band. Summing these values, we obtain the final predicted value. NHITS uses a similar architecture but adds a MaxPooling layer at the top of each block. The kernel size of these layers decreases to sample data at multiple rates. As a result, the frequency bands can be extracted in decreasing order of wavelength. In addition, during the synthesis process, NHITS uses an interpolation function to reduce the computation time of the network compared to NBEATS.

Although they can perform well even in the problem of predicting a series of future value, the above methods encounter great difficulties when solving problems on aperiodic time-series data, even in the problem of predicting the next trend of the data. This is because aperiodic time-series data exhibits very weak periodicity. Thus, it is difficult for frequency-based methods to capture this feature. In addition, the above methods also have a disadvantage similar to traditional deep learning methods. They do not have a mechanism for synthesizing multi-source information. Therefore, a new approach, which can both synthesize multi-source information and effectively capture temporal features, is needed.

2.3. Optimization-based Meta-learning

Optimization-based ML is a training method that allows model to gain experience after learning different tasks in the same task distribution. This equips the machine learning model with the ability to generalize highly and adapt quickly to new tasks after only a few training steps with limited training data. With this ability, ML is widely used in tasks that require the ability to fast adapt to new data (e.g. personalization of learning models [6, 9, 25], domain adaptation in online learning [15, 34]).

A basic ML algorithm is trained on multiple tasks t drawn from the same task distribution $\mathcal{T}$ [14]. The data for task t is divided into a support set $\mathcal{D}_t^{support}$ and a query set $\mathcal{D}_t^{query}$. During the learning process, two optimization steps, inner and outer optimization, are performed alternately. Inner optimization attempts to find an optimal set of parameters θ_t^* for each machine learning model on the support set of each task using the equation 1.

$$\theta_t^* = \theta_t(\phi) = \arg\min_{\theta} \mathcal{L}_t^{task}\left(\phi, \mathcal{D}_t^{support}\right) \tag{1}$$

In which, ϕ is the result of the outer optimization process, which acts as the initial value of θ_t. $\mathcal{L}_t^{task}$ is the error function of the model on the support set of task t.

The algorithm then uses the optimal parameter sets θ_t^* to perform on the corresponding query set. The losses of the entire models are then aggregated to perform the outer optimization process as equation 2.

$$\phi^* = \arg\min_{\phi} \sum_t \mathcal{L}_t^{meta}\left(\theta_t(\phi), \mathcal{D}_t^{query}\right) \tag{2}$$

By performing the above training method, the ϕ^* model will have a high level of generalization across different tasks, and can quickly respond to a new task after only a few training steps. In the context of aperiodic time-series data, the model can quickly learn and adapt to the new variance of data over time.

In the inference phase, the initial values for the model parameters are assigned ϕ^*. The model is then adapted quickly to the support set and performed on the query set. The results on the query set are the model output.

One disadvantage of optimization-based ML method lies in solving equation 2 which requires massive overhead to compute and maintain a Hessian matrix. Even so, ML algorithms still achieve a high accuracy in handling many problem in which the quick adaptation or effective model synthesis are required.

3. Methodology

We propose Temporal-ML, a ML-based method which consists of two main components that work in parallel: (1) - Temporal feature extraction; (2) - Models' parameter synthesis. The overview of the method is illustrated in Figure 1. In subsection *Temporal feature extraction*, we propose to use the features extracted from BiLSTM network. In subsection *Effective synthesis of model's parameters*, MAML is utilized to synthesize the parameters of the models.

Due to the contribution of BiLSTM features, we aim to effectively extract hidden temporal features from aperiodic data. By using MAML in the weight aggregation process, the proposed method is expected to be a reasonable and

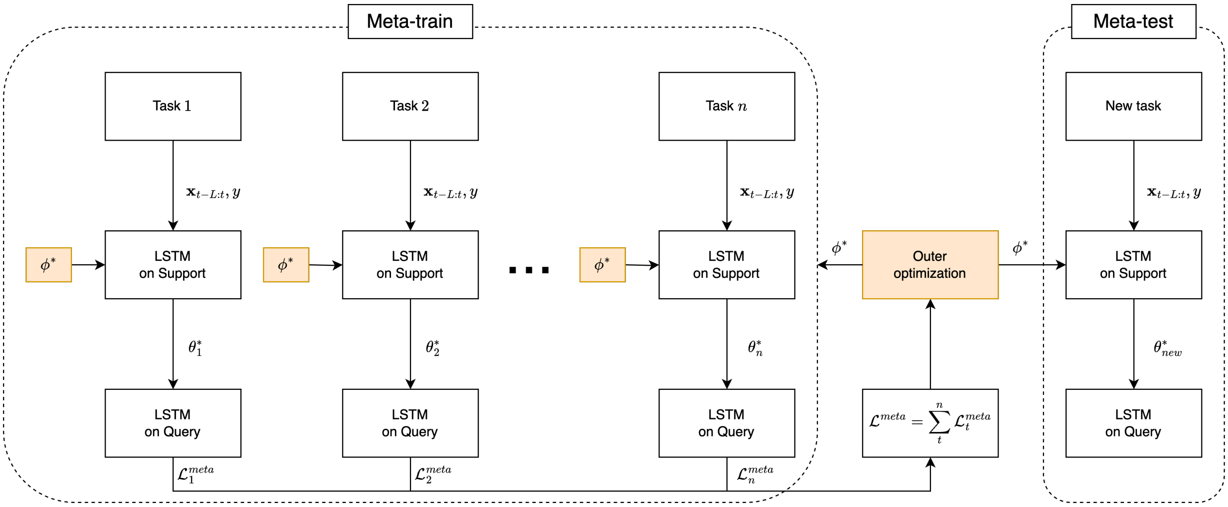

Figure 1: The flow of meta-training and meta-testing on multi-fx dataset.

effective alternative to traditional ensemble models in effectively synthesizing external factors, minimizing the impact of variance variation, and efficiency capturing hidden long-term dependencies in the past.

3.1. Data preparation

`Temporal-ML` bases on ML algorithms to train the model. Therefore, the data must be reorganized so that the ML algorithms can work. In case the data includes many different datasets belonging to the same field, each dataset will be considered a task. The goal of using multiple datasets of the same domain is to utilize the corelation between them as indicated in study [2, 24]. In case the data includes a single dataset, it is necessary to divide this dataset into subsets corresponding to separate tasks. By this way, we aim to enhance the learning of information about external factor reflected in single dataset [10]. In summary, the prepared dataset includes n tasks: $\mathcal{D} = \left\{\mathcal{D}_t\right\}_{t=1}^{n}$. The data at each task is divided into support and query sets: $\mathcal{D}_t = \left\{\mathcal{D}_t^{support}, \mathcal{D}_t^{query}\right\}$.

A datapoint consists of a pair of values $(\mathbf{x}_{t-L:t}, y)$. In which, $\mathbf{x}_{t-L:t}$ includes L historical values from timestamp t; $y \in \{0, 1\}$ is the label, showing the decreasing or increasing trend of the data sample x_{t+1} compared to x_t. Depending on each problem and the implementation, the elements in $\mathbf{x}_{t-L:t}$ can be a matrix or a vector. For example, consider stock data, $\mathbf{x}_{t-L:t}$ can contain L vectors $\vec{x}_i$ = (open, low, high, close) or can be a vector of close price values only.

3.2. Temporal feature extraction

Temporal feature extraction is performed using `BiLSTM`. Specifically, since every datapoint in time-series data usually has a small number of attribute, each element in the matrix $\mathbf{x}_{t-L:t}$ (abbreviated as $\mathbf{x}$) is passed through a `FullyConnected` layer (with a non-linear activation function) whose output is larger than the dimension of $\vec{x}_i, i \in [t-L, t]$ to obtain vector $\vec{x}'_i$ (equation 3). Accordingly, the data characteristics are expressed deeper and clearer. These features are then passed through the `BiLSTM` network (equation 4) to selectively extract long-term temporal dependencies ($\mathbf{h}_{BiLSTM}$). At the end, $\mathbf{h}_{BiLSTM}$ is passed through a classification head of the neural network (equation 5) to predict the label of $\mathbf{x}$.

$$\mathbf{x}' = \textbf{FullyConnected}\left(\mathbf{x}\right) \tag{3}$$

$$\mathbf{h}_{BiLSTM} = \textbf{BiLSTM}\left(\mathbf{x}'\right) \tag{4}$$

$$\hat{y} = \textbf{FullyConnected}\left(\mathbf{h}_{BiLSTM}\right) \tag{5}$$

`BiLSTM` maintains cell-state values to selectively store temporal dependencies, as well as to mitigate the vanishing gradient problem during training. Therefore, it is well suited for solving time-series data problems. Compared to `LSTM`, BiLSTM is superior in capturing global context of a sequence as well as solving the problem of information forgetting when dealing with long sequences in LSTM. This is because BiLSTM can extract and synthesize information from both sides of the input sequence. In conclusion, BiLSTM can capture the entire context with minimal information loss for long sequences. In the context of the movement prediction problem of aperiodic time-series data, BiLSTM is expected to successfully capture continuous value variation as well as not miss important information at distant inputs.

3.3. Efficient model synthesis

We use MAML to train and aggregate the weights of the models of all tasks via the CrossEntropyLoss (equation 6).

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbb{E}\left[(\mathbf{y}\log\hat{\mathbf{y}}) + (1-\mathbf{y})\log(1-\hat{\mathbf{y}})\right] \tag{6}$$

As mentioned in section 2.3, parameter optimization in the ML approach is to solve the two equations 1 and 2 using optimization methods on the support set and query set. Specifically, the optimization process includes many global steps (outer optimization), performed on all tasks participating in training. Each global step includes many local steps (inner optimization) performed on each individual task. At global step r, the eth local optimization process on the support set of task t proceeds as follows:

$$\begin{cases} \theta_t^{(0)} & = \phi_{r-1} \\ \theta_t^{(e)} & = \theta_t^{(e-1)} - \alpha\nabla_\theta\mathcal{L}_t^{task}\left(\theta_t^{(e-1)}, \mathcal{D}_t^{support}\right) \end{cases} \tag{7}$$

In which, ϕ_{r-1} is the result of the $r-1$ outer optimization process, α is the inner learning rate, $\mathcal{L}_t^{task}$ evaluates the use of $\theta_t^{(e-1)}$ on $\mathcal{D}_t^{support}$.

Next, the outer optimization process is performed by aggregating the losses on the query set of the tasks and optimizing on it (equation 8).

$$\begin{cases} \phi_0 = \text{Random Initialization} \\ \phi_r = \phi_{r-1} - \beta\nabla_\phi \sum_{t=1}^{n} \mathcal{L}_t^{meta}\left(\theta_t(\phi), \mathcal{D}_t^{query}\right) \end{cases} \tag{8}$$

Where, β is the outer learning rate, $\theta_t^*(\phi)$ is the local optimized weight obtained from equation 7, $\mathcal{L}_t^{meta}$ evaluates the use of $\theta_t(\phi)$ on $\mathcal{D}_t^{query}$.

Assuming the algorithm runs E steps in inner optimization, the derivative quantity at equation 8 is rewritten as follows (the notations of dataset are removed):

$$\nabla_\phi \sum_{t=1}^{n} \mathcal{L}_t^{meta}\left(\theta_t - \alpha\nabla_\theta\mathcal{L}_t^{task}\left(\theta_t\right)\right) \tag{9}$$

$$= \sum_{t=1}^{n} \frac{\partial\mathcal{L}_t^{meta}\left(\theta_t^{(E)}\right)}{\partial\theta_t^{(E)}} \frac{\partial\theta_t^{(E)}}{\partial\phi} \tag{10}$$

$$= \sum_{t=1}^{n} \nabla_\theta\mathcal{L}_t^{meta}\left(\theta_t^{(E)}\right) \prod_{j=0}^{E-1}\left[\mathbb{I} - \alpha\nabla_\theta^2\mathcal{L}_t^{task}\left(\theta_t^{(j)}\right)\right] \tag{11}$$

As obtained in equation 11, the outer optimization process requires large overhead in terms of memory and computation to train the model. As such, the number of computational steps to find θ^* needs to be limited. In practice, methods using ML [6, 9, 11, 20, 25] often choose $E \in [1, 5]$.

Under the view point of ensemble learning, the synthesis process can be regarded as an efficient way to aggregate models. Thus, information from multiple data sources is efficiently aggregated. The model can then capture the correlations between different datasets. Moreover, by dividing a dataset into multiple sequences, we ensure that the global model can learn temporal feature from entire dataset, reducing the problem of information forgetting in BiLSTM.

4. Numerical experiment

4.1. Dataset

Foreign exchange in particular and financial indices in general are typical data types for aperiodic time-series data. Therefore, we choose this type of data to test the model. Specifically, we configure two datasets using foreign exchange data. `USD/JPY` dataset consists of only data of the exchange rate between US dollar and Japanese yen. The data is sampled hourly from 2000 to 2024, including 4 attributes: *open, low, high, and close price*. By carefully examining the transaction history as well as effectively aggregating data characteristics, we expect to forecast the movement of price without utilizing external information. `multi-fx` dataset consists of 60 currency pairs made of 18 countries: Australia, Canada, Switzerland, Denmark, EU, United Kingdom, Hong Kong, Iceland, Japan, Norway, New Zealand, Singapore, Sweden, Turkey, United States, Mexico, China, South Africa. The data has similar attributes to `USD/JPY` and is sampled daily from 2014 to 2024. By integrating analysis of multiple sources of information in the domain of FX, we expect to obtain valuable analytical insights for the indicator of interest.

In addition, we use two periodic datasets: Electricity Transformer Temperature (`ETT-m2`) [40] and Weather (`WTH`) [17]. `ETT-m2` dataset consists of 7 attributes, measuring parameters of a transformer in a province of China every 15 minutes from July 2016 to July 2018. `WTH` dataset consists of 12 attributes, recording the weather parameters at the Weather Station of the Max Planck Biogeochemistry Institute in Jena, Germany every 10 minutes in 2020. These two datasets exhibit very strong periodicity, which frequency decomposition-based methods have been very successful in predicting. Experiments on them provide a more comprehensive insight of the proposed method's capabilities.

4.2. Metric

Our study uses *Accuracy, Precision, Recall*, and *F1-score* to evaluate the proposed method. Suppose `Temporal-ML` is evaluated on a dataset $\mathcal{D}$ consisting of a attributes and divided into n tasks. The evaluation is performed by letting the model predict the movement of each attribute with all attributes as input, then performing two aggregation steps: (1) - Aggregation on tasks; (2) - Aggregation on attributes. This evaluation process is designed based on study [5].

Aggregation on tasks. Consider metric m when the model predicts any attribute k ($1 \leq k \leq a$). After predicting attribute k, we obtain n values: $\{m_1^{(k)}, \ldots, m_n^{(k)}\}$. The process of synthesizing metrics m of n tasks is performed as follows:

$$\bar{m}^{(k)} = \frac{1}{n}\sum_{i=1}^{n} m_i^{(k)} \tag{12}$$

$$s_m^{(k)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(m_i^{(k)} - \bar{m}^{(k)}\right)^2} \tag{13}$$

Aggregation on attributes. The model predicts all a attributes and obtains a metrics: $\left\{\left(\bar{m}^{(k)} \pm s_m^{(k)}\right)\right\}_{k=1}^{a}$. The process of synthesizing metric m of a attributes is as follows:

$$\bar{m} = \frac{1}{a}\sum_{k=1}^{a} \bar{m}^{(k)} \tag{14}$$

$$s_m = \sqrt{\frac{1}{a}\sum_{k=1}^{a}\left(s_m^{(k)}\right)^2} \tag{15}$$

4.3. Experimental setup

This study uses `NHITS` (AAAI 2023) as the baseline model. `NHITS` uses the entire attributes to predict the next trend for an attribute in a dataset. The algorithm aims to decompose the frequency bands and combine them to predict the future. For `USD/JPY`, `ETT-m2`, and `WTH` datasets, the data in each set is divided into a training set, a validation set, and a test set with a ratio of 6:2:2, respectively. For `multi-fx` dataset, since `NHITS` does not have a mechanism to aggregate models across different datasets, we repeat the training process (training, validation, testing on 60%, 20%,

Table 1
Statistics on datasets.

Dataset	#Attribute	#Task	#Sample	#Sample/Task
USD/JPY	4	60	150,175	2,503
multi-fx	4	120	154,440	1,287
ETT-m2	7	48	69,680	1,452
WTH	12	40	35,064	877

20% of the data, respectively) for each currency pair, then aggregate metrics across currency pairs to obtain the final evaluation. In addition, a random model is also used for comparison. This model generates predicted values of 0 and 1 according to a uniform distribution. The process of synthesizing metrics of this model is similar to `NHITS`.

As mentioned in subsection 3.1, to use ML, the datasets need to be structured into separate tasks. In each task, the support set accounts for 20% of the data, the query set accounts for 80% of the data. The statistics of tasks in datasets are presented in Table 1. In which, the ratio of tasks used for meta training, validation, and testing is 50:25:25. The details of dividing tasks is presented in Appendix A.

Before data is fed to model, it is normalized by `z-score`. Specifically, training data is used to compute mean and standard deviation of each attribute. Then, these values are used to normalize validation and testing data. All models are trained on training set and validated on validation set. The best model is selected based on the performance on validation set. The final performance on testing set is reported. The details of tuning process is presented in Appendix B.

4.4. Ablation study

We evaluate each component in `Temporal-ML` by removing or replacing it and compare the results to the original algorithm. Specifically, we conduct experiments on two groups of algorithms: (1) - Model with `BiLSTM` replaced; (2) - Model without ML. Regarding the evaluation of these models, ML-based models are evaluated as `Temporal-ML`, the remaining models are evaluated as `NHITS`. The details of model architectures are presented in Appendix C.

Model with `BiLSTM` replaced. In this group, we replace `BiLSTM` with `BiLSTM+CNN`, `Transformers`. The goal of replacing `BiLSTM` with `BiLSTM+CNN` is to test the local feature extraction ability of the extractor. In order to do so, we concatenate the features of `BiLSTM` and `CNN` then performing classification. The purpose of using `Transformers` is to compare the performance of attention mechanism to `BiLSTM` on aperiodic time-series data.

Model without ML. To study the impact of `MAML` as well as optimization-based ML algorithms on `Temporal-ML`, we remove this component and only use the pure feature extraction methods introduced above: `BiLSTM`, `BiLSTM+CNN`, `Transformers`. Through this, the ability to effectively synthesize information from multiple sources of `MAML` in aperiodic time-series data will be clarified. The dataset which contains information from multiple sources is `multi-fx` dataset. Thus, we only experiment on this dataset.

5. Result & Discussion

5.1. Main result

The main results of this work are shown in Table 2. Accordingly, our proposed method outperforms `NHITS` on all metrics on two aperiodic datasets. On periodic datasets, our model achieves a competitive results compared to the baseline model. Specifically, for `USD/JPY` and `multi-fx` dataset, `Temporal-ML` improves from 12% to 14% on metrics compared to the `NHITS`. On the other hand, for periodic datasets, the accuracy of `Temporal-ML` is 1% lower, and the remaining metrics are 4-5% lower than `NHITS` on `ETT-m2` data. For strongly cyclical data such as `WTH`, our algorithm outperforms `NHITS` by about 1%. In addition, `Temporal-ML` also achieves 1-3% lower dispersion on metrics than `NHITS` when operating in multi-source data context.

In Table 2, the results are aggregated as the average of all features, so they are highly representative. However, these results lack detail in the evaluation. Therefore, we analyze the results of the algorithms on each attribute of the aperiodic datasets in Table 3. Accordingly, when `NHITS` predicts the attributes *hight, low, close price* on both datasets, the results

Table 2
Classification results (%) of Temporal-ML and NHITS on aperiodic (USD/JPY, multi-fx) and periodic (ETT-m2, WTH) datasets. Best results per metrics are boldfaced. (*): Our method.

		accuracy	*precision*	*recall*	*F1 − score*
USD/JPY	NHITS	58.46	58.24	57.65	55.82
	Temporal-ML*	**70.33 ± 1.69**	**70.73 ± 1.82**	**70.26 ± 1.93**	**69.28 ± 2.50**
multi-fx	NHITS	53.51 ± 5.02	53.90 ± 7.68	53.64 ± 4.05	50.20 ± 7.35
	Temporal-ML*	**66.26 ± 7.45**	**67.08 ± 8.97**	**65.76 ± 7.79**	**64.06 ± 10.00**
ETT-m2	NHITS	**71.88**	**66.86**	**62.49**	**62.69**
	Temporal-ML*	71.14 ± 9.13	62.21 ± 7.51	58.75 ± 5.30	57.61 ± 6.74
WTH	NHITS	74.18	68.05	65.04	65.40
	Temporal-ML*	**74.97 ± 2.63**	**69.13 ± 3.44**	**65.98 ± 3.96**	**66.00 ± 3.80**

Table 3
Accuracy (%) of Temporal-ML and NHITS on each attribute of USD/JPY and multi-fx dataset. Best results per metrics are boldfaced. (*): Our method.

		Open	*High*	*Low*	*Close*
USD/JPY	Random	49.91	49.91	50.33	50.66
	NHITS	75.03	53.74	51.57	53.50
	Temporal-ML*	**93.06 ± 1.36**	**68.87 ± 1.53**	**51.01 ± 1.58**	**68.37 ± 2.16**
multi-fx	Random	49.92 ± 2.19	49.93 ± 2.22	49.94 ± 2.17	49.85 ± 2.13
	NHITS	58.36 ± 7.13	50.66 ± 4.28	51.17 ± 3.21	53.84 ± 4.60
	Temporal-ML*	**78.13 ± 6.33**	**62.74 ± 3.30**	**58.28 ± 3.86**	**65.88 ± 12.49**

are only very close to the random prediction model. This shows that prediction on aperiodic data is really difficult. On the other hand, `Temporal-ML` fails to predict *low price* of `USD/JPY` (accuracy approaching random prediction level). All other features of both datasets achieve high convergence and are significantly higher than `NHITS`. This demonstrates the superiority of the proposed algorithm on aperiodic data.

To clarify the capabilities of `Temporal-ML`, based on the main results and the results obtained from ablation experiments (subsection 4.4), we analyze the proposed algorithm on two aspects: (1) - Temporal feature extraction ability; (2) - Models' parameters aggregation ability.

5.2. Temporal feature extraction ability

From the tables 2 and 3, it is not difficult to see that the temporal feature extraction ability of `Temporal-ML` is much better than that of `NHITS`. Indeed, the feature extraction and synthesis process of `NHITS` shows that this method is trying to simulate the frequency resolution process of Fourier transform. The frequency band features are the most important information for periodic data. Therefore, the predictions of `NHITS` can easily achieve high accuracy on this type of data. However, it is very difficult to achieve frequency resolution on aperiodic data. Furthermore, the time-series data, although filtered through multiple stacks to extract low-to-high frequency bands, the filtering techniques used in the network are extremely simple (using only `FullyConnected` with nonlinear activation functions and `Pooling` layers). This is what makes `NHITS` a major obstacle on `multi-fx`, `USD/JPY` datasets and aperiodic data in general.

To better understand the capabilities of `BiLSTM`, we performed an ablation study on this component of `Temporal-ML` and presented the results in Table 4. Accordingly, when `BiLSTM` is replaced by other components (`BiLSTM+CNN`, `Transformers`) for temporal feature extraction, the results on all four metrics of most datasets are lower than those obtained by `Temporal-ML`. Specifically, on two aperiodic datasets (`USD/JPY` and `multi-fx`), the algorithms using `BiLSTM+CNN` and `Transformers` to extract features give 2-10% lower in prediction ability than `Temporal-ML`. On periodic data, `MAML(Transformer)` still gives much lower results than `Temporal-ML`. However, `MAML(BiLSTM+CNN)` gives results that are almost asymptotically close to the proposed algorithm. On all four datasets, the standard deviation of the metrics of the proposed algorithm is generally 1-5% smaller than the remaining algorithms.

According to this result, it can be seen that using `BiLSTM` brings high and stable results in the prediction process. Using networks such as `CNN`, `Transformers` not only create negative improvements but also disturbs the learning

Table 4
Classification results (%) of Temporal-ML and models with BiLSTM replaced. Best results per metrics are boldfaced. (*): Our method.

		accuracy	*precision*	*recall*	$F1 - score$
USD/JPY	Temporal-ML*	**70.33 ± 1.69**	**70.73 ± 1.82**	**70.26 ± 1.93**	**69.28 ± 2.50**
	MAML(BiLSTM+CNN)	68.75 ± 1.78	69.15 ± 1.55	68.73 ± 1.77	69.14 ± 2.17
	MAML(Transformers)	60.67 ± 4.09	59.74 ± 6.24	59.32 ± 3.79	53.15 ± 6.19
multi-fx	Temporal-ML*	**66.26 ± 7.45**	**67.08 ± 8.97**	**65.76 ± 7.79**	**64.06 ± 10.00**
	MAML(BiLSTM+CNN)	66.08 ± 8.19	66.56 ± 9.23	65.41 ± 8.66	63.78 ± 10.83
	MAML(Transformers)	56.03 ± 6.08	56.32 ± 6.01	55.63 ± 5.01	52.33 ± 7.08
ETT-m2	Temporal-ML*	71.14 ± 9.13	62.21 ± 7.51	58.75 ± 5.30	57.61 ± 6.74
	MAML(BiLSTM+CNN)	**71.40 ± 9.10**	**63.25 ± 8.82**	**60.22 ± 8.42**	**59.43 ± 9.24**
	MAML(Transformers)	68.69 ± 8.85	58.54 ± 4.38	57.85 ± 4.71	57.07 ± 5.14
WTH	Temporal-ML*	**74.97 ± 2.63**	**69.13 ± 3.44**	**65.98 ± 3.96**	**66.00 ± 3.80**
	MAML(BiLSTM+CNN)	74.41 ± 2.71	68.17 ± 4.60	65.50 ± 4.82	65.63 ± 4.47
	MAML(Transformers)	72.28 ± 3.06	62.35 ± 8.73	60.90 ± 4.97	58.65 ± 4.72

process or reduces the efficiency of the whole model. For using BiLSTM+CNN, the learning process is disturbed because BiLSTM itself has selected effective time dependencies during the extraction process while CNN only extracts features but does not select. This leads to the instability of BiLSTM+CNN's features. If CNN extracts good features, the network will perform well (e.g., metrics on ETT-m2). On the contrary, the network will perform worse than using only BiLSTM (e.g., metrics on USD/JPY, multi-fx, WTH).

With the above analysis, we conclude that BiLSTM is not only a suitable choice for exploring the variability of aperiodic time-series data but also a suitable algorithm to combine with ML algorithms when working on time-series data in general. By using BiLSTM, we have effectively extracted market fluctuations and predicted the future.

5.3. Models' parameters aggregation ability

In the process of synthesizing features to make the final prediction, NHITS only uses simple addition and subtraction. This makes sense in the context of synthesizing information based on the frequency bands extracted from the input data. Indeed, data related to weather, demand for airplanes or electricity show very strong periodic characteristics because weather and human consumption behavior, although they may vary over time, still show a very clear seasonality (e.g., cold weather in winter, high demand for airplanes during holidays). Therefore, the frequency of this type of data is a good feature and can be easily extracted. After decomposing the input data into frequency bands, we can simply synthesize them to be able to predict the future. In addition, synthesizing the frequency bands by an addition can be considered very simple compared to current deep learning algorithms, which work on the hidden feature space of the data. Therefore, the synthesis process of NHITS on aperiodic data encounters many difficulties because the input information for the process is not good and the synthesis method is ineffective. Consequently, it can be seen that aperiodic time-series data becomes a fatal weakness for NHITS and frequency decomposition-based methods in general.

Dealing with the complexity of non-periodic data requires a complex structure that can efficiently aggregate features. As mentioned in the Introduction, analyzing and synthesizing information from multiple data sources in the same domain can provide useful information for predicting metrics of interest. Temporal-ML with the ability to synthesize information based on the optimization process of ML algorithms, allows the model to capture multi-dimensional information from different data sources, which can enhance the effectiveness of predictions.

To evaluate the capabilities of ML algorithms, we remove this component from the proposed method and then run experiments to compare with the original results. We use ML algorithms for the purpose of synthesizing multi-source information. Therefore, comparisons are only performed on the multi-fx dataset because this dataset contains several sub-datasets from different sources, which helps to accurately evaluate the information synthesis capability. The results are summarized in Table 5. Accordingly, with the information synthesis capability, Temporal-ML achieves 2-12% higher in all metrics than algorithms that do not use ML in testing process. In addition, the dispersion of the metrics is also lower from 1-7%, proving that the difference in the task prediction process is much better than the remaining algorithms.

Table 5
Classification results (%) of Temporal-ML and models without MAML on multi-fx dataset. Best results per metrics are boldfaced. (*): Our method.

	accuracy	*precision*	*recall*	*F*1 − *score*
Temporal-ML*	$\mathbf{66.26 \pm 7.45}$	$\mathbf{67.08 \pm 8.97}$	$\mathbf{65.76 \pm 7.79}$	$\mathbf{64.06 \pm 10.00}$
BiLSTM	63.88 ± 9.00	64.18 ± 13.39	63.66 ± 9.13	60.07 ± 13.09
BiLSTM+CNN	62.23 ± 8.63	62.89 ± 9.76	62.20 ± 8.34	59.91 ± 10.97
Transformers	58.44 ± 9.15	56.71 ± 15.10	58.22 ± 8.84	52.22 ± 13.12

The better performance of `Temporal-ML` on `multi-fx` is understandable because all the remaining methods in the 5 table do not have the ability to synthesize information like the proposed algorithm, but rely entirely on past data to predict the future. Therefore, not only `BiLSTM`, `BiLSTM+CNN` but also `Transformers` cannot take advantage of the correlation between data sources.

Therefore, it can be seen that, the use of optimization-based ML algorithms in information synthesis is completely reasonable because it successfully takes advantage of the correlation between data sets and achieves high accuracy without requiring too much data as in the single-source context.

6. Conclusion

Problems on aperiodic time-series data pose a great challenge to current machine learning models because this data depends on both transaction history and external factors such as economic and political situations; the variance of the data changes continuously; the aperiodicity of the data is very strong. By utilizing the time constraint extraction capability of `BiLSTM` and the efficient parameter synthesis capability of `MAML`, we propose the `Temporal-ML` algorithm with the ability to extract hidden time features as well as synthesize information from multi-source data. Based on experiments, in the process of solving the next trend prediction problem, the proposed method achieves high efficiency on two aperiodic data sets (`USD/JPY` and `multi-fx`) and equivalent efficiency on periodic data sets (`ETT-m2` and `WTH`) compared to `NHITS` - a frequency decomposition based approach. In addition, by performing an ablation study, we demonstrate the superior of `BiLSTM` in extracting temporal feature compared to `CNN`, `Transformers` and the effectiveness of `MAML` in synthesizing information from multi-source data. The results of this study are expected to be a good foundation for future research on time-series data.

Our experiment only illustrates a typical case of efficient model synthesis. In the future, by using different ML algorithms such as `Meta-SGD` [20], `Reptile` [26], `iMAML` [28], it is possible to generate new models with higher accuracy. On the other hand, it is possible to extend this method to solve long-horizon prediction problems. Indeed, by changing the output and error of the model, it is possible to solve these problems. However, the architecture of the sub-models needs to be re-examined to better suit the new problem.

A. Data preprocessing

A.1. Task-based preprocessing

Since the datasets are all time-series, the training data should not reveal any information about the future. Therefore, we divide every currency pair in `multi-fx` into 2 tasks, resulting in 120 tasks. The first parts of every pair are used for meta-training. The second parts of them are used for meta validating and meta testing with the ratio of 50:50. The remaining datasets are is divided into periods of time, each corresponding to a task.

The `z-score` preprocessing is performed on training data to obtain mean and standard deviation values. These values are used to normalize the data of validation set and test set.

A.2. Fairness in evaluation

Let m be the number of data samples in each task, the total data of training, validating, and testing is: mn. For `NHITS`, the model is trained, validated, and tested on 60%, 20%, 20% of the data, respectively:

$$\text{Train: } 0.6mn \tag{16}$$

Table 6
Search space for fine-tuning our method.

Hyper-parameter	Seach space
Inner batch size (samples/batch)	{32}
Inner training step	{3}
Outer batch size (tasks/batch)	{5}
Outer training step	{100}
Lookback window	{10, 20, 30}
Inner learning rate	{0.001, 0.00325, 0.0055, 0.00775, 0.01}
Outer learning rate	{0.001, 0.00325, 0.0055, 0.00775, 0.01}

$$\text{Validation: } 0.2mn \tag{17}$$

$$\text{Testing: } 0.2mn \tag{18}$$

For `Temporal-ML`, the knowledge enrichment process for the model uses the entire training data and the support set of the testing/validation data. The testing/validation process is performed on the query set:

$$\text{Train: } 0.5mn + 2 \times 20\%(0.25mn) = 0.6mn \tag{19}$$

$$\text{Validation: } 80\%(0.25mn) = 0.2mn \tag{20}$$

$$\text{Testing: } 80\%(0.25mn) = 0.2mn \tag{21}$$

Therefore, with the above data division, we achieve fairness in the number of data samples during testing.

B. Hyperparameter tuning

B.1. Temporal-ML

In our implementation, we use a `FullyConnected` layer of 16 units with a `ReLU` activation function to represent deeper and more explicit features. This feature is then passed to a `BiLSTM` block. The `BiLSTM` block consists of 32 hidden units, two last outputs of which are concatenated to form a final vector. This vector is then passed through a binary classification layer with a `Sigmoid` activation function.

The fine-tuning process of ML algorithms involves many hyper-parameters such as inner batch size, outer batch size, inner training step, outer training step, etc. To make fine-tuning easy, we fix most of the parameters and only fine-tune the size of lookback window, inner and outer learning rate. Details are presented in Table 6.

B.2. NHITS

We rely on study [5] to define the search space (Table 7) for parameter fine-tuning. For parameters not mentioned in the table, we use the default values of the `NHITS` implementation in `NeuralForecast` library [12]. The best results of fine-tuning process are selected and reported in this study.

C. Model architecture in ablation study

`BiLSTM+CNN` model includes two parts: `BiLSTM` and `CNN`. The `BiLSTM` part has the similar architecture with the one used in `Temporal-ML`. The `CNN` part consists of two `Convolution1D` layers with `ReLU` as activation function and the number of filters are 32 and 64, respectively. Each of convolution layer is followed by a `MaxPooling` layer with the kernel size of 2. Next, the output of `CNN` part is flatten and concatenated with the output of the `BiLSTM` part. The concatenated is then passed through a classification head with `Sigmoid` activation function.

`Transformer` model is composed of two encoders. Each encoder uses three attention heads, and the key vector dimension is 256. The `FeedForward` part of the encoder uses two `Convolution1D` layers and one `LayerNormalization`. After stacking the two encoder layers, the feature matrix is passed through a `Flatten` layer and then fed into the classification head.

Table 7
Search space for fine-tuning NHITS.

Hyper-parameter	Seach space
Random seed	{1}
Number of stacks	{3}
Number of blocks in each stack	{1}
Activation function	{ReLU}
Batch size	{256}
Epoch	{500}
Lookback window	{10, 20, 30}
Pooling kernel	{[2,2,2], [4,4,4], [8,8,8], [8,4,1], [16,8,1]}
Stacks' coefficients	{[168,24,1], [24,12,1], [180,60,1],[40,20,1], [64,8,1]}
Number of MLP layers	{1,2}
Learning rate	{0.001, 0.00325, 0.0055, 0.00775, 0.01}

CRediT authorship contribution statement

Bao Long Nguyen: Data curation, Methodology, Programming, Writing - Original draft preparation. **Kenta Hongo:** Literature review. **Ryo Maezono:** Conceptialization. **Bac Le:** Review paper. **Tom Ichibha:** Supervision.

References

[1] Ali, M., Prasad, R., Xiang, Y., Yaseen, Z.M., 2020. Complete ensemble empirical mode decomposition hybridized with random forest and kernel ridge regression model for monthly rainfall forecasts. Journal of Hydrology 584, 124647.

[2] Andraw W. Lo, A.C.M., 1990. When are contrarian profits due to stock market overreaction? The review of financial studies 3, 175–205.

[3] Ayitey Junior, M., Appiahene, P., Appiah, O., Bombie, C.N., 2023. Forex market forecasting using machine learning: Systematic literature review and meta-analysis. Journal of Big Data 10, 9.

[4] Bahdanau, D., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .

[5] Challu, C., Olivares, K.G., Oreshkin, B.N., Ramirez, F.G., Canseco, M.M., Dubrawski, A., 2023. Nhits: Neural hierarchical interpolation for time series forecasting, in: Proceedings of the AAAI conference on artificial intelligence, pp. 6989–6997.

[6] Chen, F., Luo, M., Dong, Z., Li, Z., He, X., 2018. Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876 .

[7] Chen, M., Peng, H., Fu, J., Ling, H., 2021. Autoformer: Searching transformers for visual recognition, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 12270–12280.

[8] Cui, Z., Ke, R., Pu, Z., Wang, Y., 2018. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143 .

[9] Fallah, A., Mokhtari, A., Ozdaglar, A., 2020. Personalized federated learning: A meta-learning approach. arXiv preprint arXiv:2002.07948 .

[10] Fama, E.F., 1970. Efficient capital markets: A review of theory and empirical work. Journal of finance 25, 383–417.

[11] Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks, in: International conference on machine learning, PMLR. pp. 1126–1135.

[12] Garza, A., . Neural forecast - user friendly state-of-the-art neural forecasting models. URL: https://github.com/Nixtla/neuralforecast.

[13] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

[14] Hospedales, T., Antoniou, A., Micaelli, P., Storkey, A., 2021. Meta-learning in neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence 44, 5149–5169.

[15] Hu, N., Mitchell, E., Manning, C.D., Finn, C., 2023. Meta-learning online adaptation of language models. arXiv preprint arXiv:2305.15076 .

[16] Koç, E., Koç, A., 2022. Fractional fourier transform in time series prediction. IEEE Signal Processing Letters 29, 2542–2546.

[17] Kolle, O., 2008. Weather dataset. URL: https://www.bgc-jena.mpg.de/wetter/.

[18] Lai, G., Chang, W.C., Yang, Y., Liu, H., 2018. Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st international ACM SIGIR conference on research & development in information retrieval, pp. 95–104.

[19] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., 1989. Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems 2.

[20] Li, Z., Zhou, F., Chen, F., Li, H., 2017. Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 .

[21] Liu, H., Wang, Z., Dong, X., Du, J., 2025. Onsitnet: A memory-capable online time series forecasting model incorporating a self-attention mechanism. Expert Systems with Applications 259, 125231.

[22] Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., Xu, Q., 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. Advances in Neural Information Processing Systems 35, 5816–5828.

[23] Luong, M.T., 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 .

[24] Mech, T.S., 1993. Portfolio return autocorrelation. Journal of Financial Economics 34, 307–344.

[25] Nguyen, B.L., Cao, T.C., Le, B., 2022. Meta-learning and personalization layer in federated learning, in: Asian Conference on Intelligent Information and Database Systems, Springer. pp. 209–221.

[26] Nichol, A., Achiam, J., Schulman, J., 2018. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 .

[27] Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y., 2019. N-beats: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437 .

[28] Rajeswaran, A., Finn, C., Kakade, S.M., Levine, S., 2019. Meta-learning with implicit gradients. Advances in neural information processing systems 32.

[29] RB, C., 1990. Stl: A seasonal-trend decomposition procedure based on loess. J Off Stat 6, 3–73.

[30] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. nature 323, 533–536.

[31] Sadeghi, A., Daneshvar, A., Zaj, M.M., 2021. Combined ensemble multi-class svm and fuzzy nsga-ii for trend forecasting and trading in forex markets. Expert Systems with Applications 185, 115566.

[32] Samarawickrama, A., Fernando, T., 2019. Multi-step-ahead prediction of exchange rates using artificial neural networks: a study on selected sri lankan foreign exchange rates, in: 2019 14th Conference on Industrial and Information Systems (ICIIS), IEEE. pp. 488–493.

[33] Shih, S.Y., Sun, F.K., Lee, H.y., 2019. Temporal pattern attention for multivariate time series forecasting. Machine Learning 108, 1421–1441.

[34] Son, J., Lee, S., Kim, G., 2024. When meta-learning meets online and continual learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence .

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

[36] Vettoruzzo, A., Bouguelia, M.R., Vanschoren, J., Rognvaldsson, T., Santosh, K., 2024. Advances and challenges in meta-learning: A technical review. IEEE Transactions on Pattern Analysis and Machine Intelligence .

[37] Zafeiriou, T., Kalles, D., 2020. Intraday ultra-short-term forecasting of foreign exchange rates using an ensemble of neural networks based on conventional technical indicators, in: 11th Hellenic Conference on Artificial Intelligence, pp. 224–231.

[38] Zhang, Y., Yan, J., 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting, in: The eleventh international conference on learning representations.

[39] Zhao, Y., Khushi, M., 2020. Wavelet denoised-resnet cnn and lightgbm method to predict forex rate of change, in: 2020 International Conference on Data Mining Workshops (ICDMW), IEEE. pp. 385–391.

[40] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W., 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI conference on artificial intelligence, pp. 11106–11115.

[41] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., Jin, R., 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International conference on machine learning, PMLR. pp. 27268–27286.