Meta-learning in movement prediction problem of aperiodic time-series data

Bao-Long Nguyen,[1, *] Kenta Hongo,[2, †] Ryo Maezono,[1, ‡] and Tom Ichibha[1, §]
[1]*School of Information Science, JAIST, Ishikawa, Japan*
[2]*Research Center for Advanced Computing Infrastructure, JAIST, Ishikawa, Japan*
(Dated: October 4, 2024)

Abstract. Predicting aperiodic time-series data (e.g. stock price, foreign exchange, Bitcoin price,...) is a difficult task for machine learning models because this type of data has high and not stationary variance; does not show clearly periodicity, making it difficult to extract features; depends not only on past values but also on external factors, which make them unstable and non-cyclical, such as economic and political situations. To overcome the above challenges, we use Meta-learning to train a combined `LSTM` and `CNN` network, thereby effectively extracting and synthesizing hidden features of the data over time. Experiments on foreign exchange data of 60 currency pairs over 24 years (2000-2024) show that the proposed method performs well and has 5%-11% higher accuracy compared to the `NHITS` - the state-of-the-art model in 2023 on time-series data, in the task of predicting the trend (upward or downward) of the next trading day.

I. INTRODUCTION

Aperiodic time-series prediction in general or foreign exchange (FX) prediction in particular has been a matter of concern for many researchers [13, 17, 22] for a long time. The two main techniques used in aperiodic time-series prediction are fundamental analysis and technical analysis [3]. Fundamental analysis focuses on analyzing external factors such as policies and economic strategies of companies and countries to predict the future. Meanwhile, technical analysis relies entirely on historical value fluctuations, which are difficult to capture external factors, to analyze future trends.

Aperiodic time-series data prediction faces several inherent challenges: (1) - The variance varies significantly over time, making it difficult to assume a specific distribution to approximate errors, which is a challenge for machine learning models in accurately predicting future values and trends. (2) - Aperiodic data does not follow explicit patterns, making it challenging to identify the latent features required for reliable predictions. (3) - Aperiodic time-series data is influenced not only by historical data but also by numerous external factors [22]. For example, a company's stock price is affected by news, economic conditions, political events, and other external influences.

For the variance variation challenge, ensemble models [1, 29, 34] are often used to mitigate the effects of variance variation. Ensemble models provide a holistic and multi-perspective view based on sub-models, thereby helping the overall model adapt to strong variance changes. However, under the rigid synthesis of traditional ensemble models, the results obtained on non-periodic datasets are very poor.

To overcome the pattern variation challenge, most studies use features extracted from Long Short-Term Memory neural network (`LSTM`) [14], Artificial Neural Network (`ANN`), and Convolution Neural Network (`CNN`) [21]. Specifically, in 2022, 20% of all publications related to financial index prediction used `LSTM`, 20% used `ANN`, and 6% used `CNN` [3]. This is completely understandable because `LSTM` and `CNN` have proven their ability to extract spatial and temporal features effectively. However, when faced with aperiodic data with only 2600 samples, after 40 training steps, `LSTM` is still underfitting, while `CNN` is overfitting after only 5 training steps.

Regarding the external influences challenge, [9] hypothesizes that different time-series datasets of the same domain at the same time reflect the impact of external factors. For example, studies [2, 24] show the dependence between the financial ratios of a given company and the ratios of other companies. This further strengthens the hypothesis in study [9]. Therefore, by exploiting data sources with the same domain, we can capture information reflecting external factors.

We approach the above challenges through combination of feature combination and model synthesis. Specifically, by using Meta-learning (ML) [10], we efficiently synthesize the parameters of local models, which significantly reduces variance loss and increases the ability to collect information reflecting external factors from datasets in the same domain. To effectively learn laten patterns, we propose a method that combines features from `CNN` and `LSTM`. In addition, we assume that aperiodic time-series data also has hidden long-term dependencies. Traditional approaches use a fixed amount of past data (lookback window) to train the model. This poses a major obstacle to the learning process because long-term features will be forgotten over time. On the other hand, ML divides the dataset into many parts to learn and synthesize the learned parameters effectively, so it can handle this challenge well.

Finally, we demonstrate the superiority of the pro-

* mwklng2309@icloud.com
† kenta_hongo@mac.com
‡ rmaezono@mac.com
§ ichibha@icloud.com

posed algorithm by solving the problem of predicting the trend (upward or downward) of foreign exchange rates and comparing the results with the existing state-of-the-art (SOTA) model (`NHITS` [4]) on two types of data: (1) - USD/JPY exchange rate data; (2) - Exchange rate data of 60 currency pairs, comprising 18 countries. These data sets are publicly available on the Internet and can be easily downloaded. The official implementation can be found at `https://github.com/baolongnguyenmac/multi_fx`.

In summary, our main contributions include the following: First, we present a novel **feature combination technique** for aperiodic time-series data that extracts and combines features using both `LSTM` and `CNN` models. Second, we uncover hidden long-term dependencies, demonstrating experimentally that aperiodic time-series data not only rely on external factors but also exhibit dependencies with past period, which we call **hidden long-term dependencies**. Third, we introduce an **efficient model parameter aggregation approach** by employing ML algorithm in place of traditional ensemble models for combining parameters from multiple machine learning models. Finally, we validate our method through experiments on exchange rate datasets and show its effectiveness in comparison to the SOTA model, `NHITS`.

II. RELATED WORKS

A. LSTM & CNN model

As aforementioned, `LSTM` is a well-known neural network for handling prediction problems on time-series data. `LSTM` is commonly used because it handles the problem of vanishing gradients well (easily encountered when using Recurrent neural networks) and can effectively exploit nonlinear relationships in data. Indeed, by maintaining the cell-state in each iteration, `LSTM` can overcome the vanishing gradient problem, thereby preserving the ability to capture long-term dependencies [6]. In addition, `LSTM` performs feature extraction with nonlinear activation functions, which helps the model parameters capture the nonlinearity of the data [12]. These factors make `LSTM` to be the first choice to think of when solving problems on time-series data.

`CNN` is widely used in image processing tasks [25, 30] because of its ability to synthesize local relationships. Not only that, `CNN` is also widely used in time-series data processing tasks such as speech recognition [7], natural language processing [31]. This proves the ability of `CNN` in discovering local temporal relationships between data samples. However, `CNN` is rarely used in aperiodic time-series data prediction tasks. In this study, we take advantage of `CNN`'s excellent local feature extraction ability to incorporate more hidden information into the model training process.

B. Optimization-based Meta-learning

Optimization-based Meta-learning (ML) algorithms, typically MAML [10] are known for their ability to train a highly general, adaptive model on new datasets with a limited amount of data and a small number of training steps [15, 32]. With this ability, ML is widely used in tasks that require the model's ability to adapt to the data (e.g. personalization of learning models [5, 8, 26], domain adaptation in online learning [16, 19]).

A basic ML algorithm is trained on multiple tasks t drawn from the same task distribution $\mathcal{T}$ [15]. The data for task t is divided into a support set $\mathcal{D}_t^{support}$ (usually small, around 20%) 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.

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

Where, 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.

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

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 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.

Hybrid ensemble models have been widely used in time-series processing problems and have been experimentally proven to be more accurate than standard time-series models because they can synthesize the strengths of many sub-models [3]. However, current ensemble model synthesis forms are still very rigid because they can only synthesize based on the final results (e.g. voting mechanism of bagging models) and near-final results (e.g. stacking models). From the perspective of ensemble models, the equation 2 can be considered an effective method of synthesizing sub-models, which helps to take advantage of the feature extraction capabilities of each model.

In other words, the synthesized model can extract features at a deeper level, significantly improving the prediction ability compared to traditional ensemble models.

C. Neural Hierarchical Interpolation for Time Series (NHITS)

`NHITS` is designed to target the prediction of long-horizon time-series data. According to [4], the structure of `NHITS` consists of multiple consecutive stacks. Each stack consists of multiple consecutive blocks. At each block, historical data is used to predict future data and past data. The residual of the previous block is used as input data for the following block. Specifically, at block l, with L past data samples ($\mathbf{y}_{t-L:t,l-1}$), the features will be extracted as follows (from [4]):

$$\mathbf{y}^{(p)}_{t-L:t,l} = \mathbf{Pooling}\left(\mathbf{y}_{t-L:t,l-1}\right) \tag{3}$$

$$\theta^b_l = \mathbf{FullyConnected}^b\left(\mathbf{y}^{(p)}_{t-L:t,l}\right) \tag{4}$$

$$\theta^f_l = \mathbf{FullyConnected}^f\left(\mathbf{y}^{(p)}_{t-L:t,l}\right) \tag{5}$$

$$\hat{\mathbf{y}}_{t-L:t,l} = g\left(\theta^b_l\right) \tag{6}$$

$$\hat{\mathbf{y}}_{t+1:t+H,l} = g\left(\theta^f_l\right) \tag{7}$$

Accordingly, `Pooling` compresses the original data into $\mathbf{y}^{(p)}_{t-L:t,l}$. `FullyConnected` contains many perceptron layers with nonlinear activation functions. θ^f_l, θ^b_l are the forecast and backcast interpolation coefficients, which are used to aggregate the output values of block l using the interpolation function $g(\cdot)$. The dimensionality of the interpolation coefficients in each stack is determined by the expressiveness ratio: $|\theta_l| = \lceil r_l H \rceil$. Typically, the expressiveness ratio will be very small in the first stacks and increase towards the end. Accordingly, the stacks can simulate frequencies from low to high. In addition, by using $g(\cdot)$, `NHITS` does not require too much hardware to train the neural network in the case of large H.

The output of block l is the forecast value $\hat{\mathbf{y}}_{t+1:t+H,l}$ and the backcast value $\hat{\mathbf{y}}_{t-L:t,l}$. The input of block $l+1$ is calculated according to the equation 8.

$$\mathbf{y}_{t-L:t,l+1} = \mathbf{y}_{t-L:t,l-1} - \hat{\mathbf{y}}_{t-L:t,l} \tag{8}$$

Suppose the model consists of S stacks, each stack has B blocks. Summing the forecast values of the blocks as in equation 9, we get the forecast value of a stack. The last residual of the last block of a stack is the input for the next stack. Finally, summing the forecast values of the stacks as in equation 10, we get the predicted forecast value of the entire network.

$$\hat{\mathbf{y}}^s_{t+1:t+H} = \sum_{l=1}^{B} \hat{\mathbf{y}}_{t+1:t+H,l} \tag{9}$$

$$\hat{\mathbf{y}}_{t+1:t+H} = \sum_{s=1}^{S} \hat{\mathbf{y}}^s_{t+1:t+H} \tag{10}$$

By concatenating stacks, each receiving the remainder of the previous stack, the above architecture is expected to decompose the data into different frequency bands (weekly, daily, even hourly). In practice, `NHITS` performs very well for highly periodic datasets such as electricity consumption, weather, traffic. However, we are aiming for an aperiodic time-series dataset, which has very low, or even non-existent, periodicity (see figure 1). This poses a huge challenge for `NHITS`.

FIG. 1: Exchange rate (close price) between US dollar and Japanese yen by day (2014-2024).

III. METHODOLOGY

Our method consists of two main parts that work in parallel: (1) - Feature extraction; (2) - Parameter synthesis. The overview of the method is illustrated in 2. In the feature extraction part, we combine two types of features from `CNN` and `LSTM` networks. In the parameter synthesis part, we use `MAML` to synthesize the parameters of the models. Due to the contribution of `LSTM` and `CNN` features, we expect to effectively extract hidden features from aperiodic data. By using `MAML` in the weight synthesis process, the proposed method is expected to be a reasonable and effective alternative to traditional ensemble models in minimizing the impact of variance variation, effectively synthesizing external factors, and preserving hidden long-term dependencies in the past.

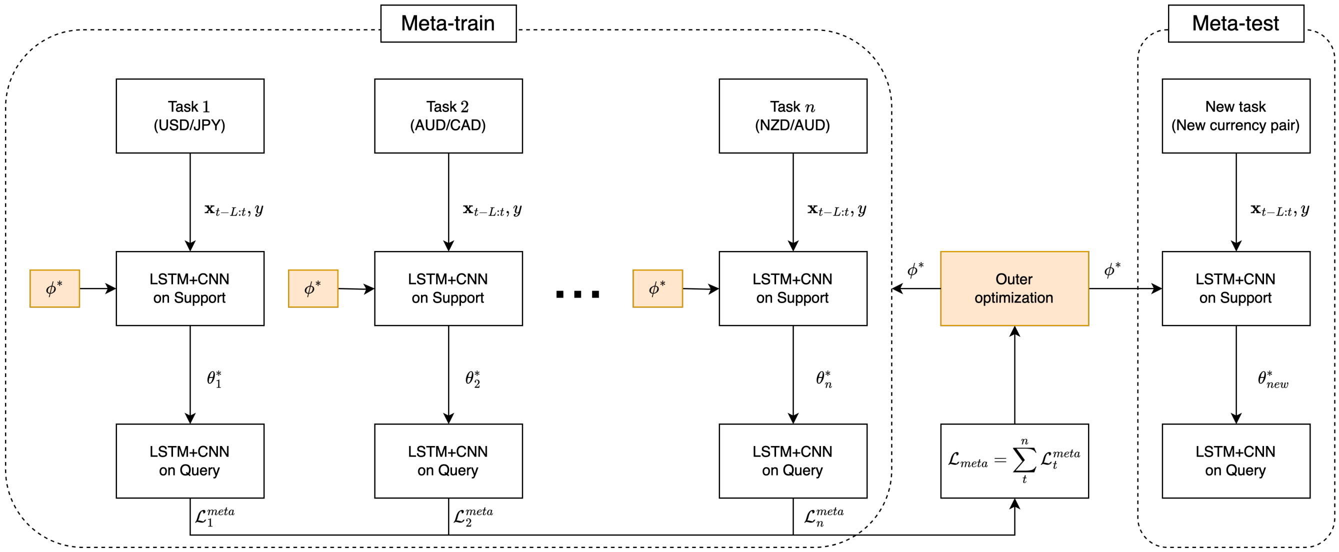

FIG. 2: The full-flow of meta-training and meta-testing on multi-fx data. Each currency pair is regarded as a task.

A. Data preparation

The proposed method uses ML algorithms to train the model. Therefore, the data needs to 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 of MAML. In case the data includes a single dataset, it is necessary to divide this dataset into subsets corresponding to separate tasks. In summary, the prepared dataset includes n tasks: $\mathcal{D} = \{\mathcal{D}_t\}_{t=1}^n$. 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 data sample consists of pairs of values $(\mathbf{x}_{t-L:t}, y)$. In which, $\mathbf{x}_{t-L:t}$ includes L historical values from time t back; $y \in \{0, 1\}$ is the data 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 vectors or scalar numbers. For example, for stock data, $\mathbf{x}_{t-L:t}$ can contain L data vectors $\vec{x}_i = $ (open, low, high, close) or just a single close price value.

B. Feature extraction

Inspired by the [33] study, we propose to combine the features extracted from LSTM and CNN networks. Specifically, we pass each element in the vector $\mathbf{x}_{t-L:t}$ through FullyConnected layers whose output'dimension is larger than the one of $\vec{x}_i, i \in [t-L, t]$ to decompose it into smaller features $\vec{x}'_i$. These features are then passed through LSTM and CNN networks to extract long-term temporal dependencies ($\mathbf{h}_{LSTM}$) and local temporal features ($\mathbf{h}_{CNN}$), respectively. To exploit the long-term temporal constraints, we use BidirectionalLSTM to extract from both sides of $\mathbf{x}_{t-L:t}$. The entire feature extraction process is summarized as follows:

$$\mathbf{x'}_{t-L:t} = \textbf{FullyConnected}\left(\mathbf{x'}_{t-L:t}\right) \tag{11}$$
$$\mathbf{h}_{LSTM} = \textbf{BidirectionalLSTM}\left(\mathbf{x'}_{t-L:t}\right) \tag{12}$$
$$\mathbf{h}_{CNN} = \textbf{Convolution1D}\left(\mathbf{x'}_{t-L:t}\right) \tag{13}$$

The LSTM network maintains cell-state values to selectively store long-term dependencies. This is very suitable for solving time-series data problems. On the other hand, future values often depend heavily on recent historical values. We propose to use the CNN network to emphasize local features, thereby directing part of the model's attention to certain time points. Therefore, the proposed method can not only remember long-term features but also highlight short-term features.

Next, $\mathbf{h}_{LSTM}$ and $\mathbf{h}_{CNN}$ are concatenated (equation 14) and then passed to the classification part of the neural network (equation 15).

$$\mathbf{h}_{t-L:t} = \textbf{Concatenate}\left(\mathbf{h}_{LSTM}, \mathbf{h}_{CNN}\right) \tag{14}$$
$$\hat{y} = \textbf{FullyConnected}\left(\mathbf{h}_{t-L:t}\right) \tag{15}$$

C. Effective synthesis of models' parameters

We use MAML to train and aggregate the weights of the models at the tasks. As mentioned in II, parameter optimization in the ML approach is to solve the two equations 1 and 2 using optimization methods on the support and query data. 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-th, the e-th local optimization process at the support set of task t occurs as follows:

$$\begin{cases} \theta_t^{(0)} & = \phi_{r-1} \\ \theta_t^{(e)} & = \theta_t^{(e-1)} - \alpha \nabla_\theta \mathcal{L}_t^{task} \left(\theta_t^{(e-1)}, \mathcal{D}_t^{support} \right) \end{cases} \tag{16}$$

In which, ϕ_{r-1} is the result of the $r-1$ global optimization process, α is the inner learning rate.

Next, the outer optimization process at the global step is performed by aggregating the losses on the query set of the tasks and optimizing on it (equation 17).

$$\begin{cases} \phi_0 = \text{Random Initialization} \\ \phi_r = \phi_{r-1} - \beta \nabla_\phi \sum_{t=1}^{n} \mathcal{L}_t^{meta} \left(\theta_t^*(\phi), \mathcal{D}_t^{query} \right) \end{cases} \tag{17}$$

Where, β is the outer learning rate.

Assuming the algorithm runs E steps in inner optimization, the derivative quantity at equation 17 is rewritten as follows (the notations of dataset are removed):

$$\begin{aligned} & \beta \nabla_\phi \sum_{t=1}^{n} \mathcal{L}_t^{meta} \left(\theta_t - \alpha \nabla_\theta \mathcal{L}_t^{task} \left(\theta_t \right) \right) \\ = & \beta \sum_{t=1}^{n} \frac{\partial \mathcal{L}_t^{meta} \left(\theta_t^{(E)} \right)}{\partial \theta_t^{(E)}} \frac{\partial \theta_t^{(E)}}{\partial \phi} \\ = & \beta \sum_{t=1}^{n} \nabla_\theta \mathcal{L}_t^{meta} \left(\theta_t^{(E)} \right) \prod_{j=0}^{E-1} \left[\mathbb{I} - \alpha \nabla_\theta^2 \mathcal{L}_t^{task} \left(\theta_t^{(j)} \right) \right] \end{aligned} \tag{18}$$

The presence of the product of second order derivatives in the equation 18 makes the derivation process complicated because it requires a lot of overhead to maintain the Hessian matrices. Therefore, the number of computational steps to find θ^* needs to be limited. In practice, methods using ML [5, 8, 10, 23, 26] often choose $E \in [1, 5]$.

IV. NUMERICAL EXPERIMENT

A. Dataset & Metric

FX 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 FX data. `USD/JPY` dataset consists of only data of the exchange rate between US dollar and Japanese yen, is divided into 60 time-sequenced subsets of equal size. The data is sampled hourly from 2000 to 2024, including the attributes of open, low, high, and close price. `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 sampled daily from 2014 to 2024.

Our goal is to use `multi-fx` dataset to extract and aggregate information about outliers (i.e. market, economic, political, etc.) that are believed to influence the outcome of a given financial index [2, 9, 24]. Meanwhile `USD/JPY` dataset is used to test our hypothesis that future data implicitly depend on certain points in the past and that efficient past feature aggregation is needed to reveal these dependencies.

In addition, we used two periodic datasets: Electricity Transformer Temperature (`ETT-m2`) [35] and Weather (`WTH`) [20]. `ETT-m2` dataset consists of 7 data fields, measuring the parameters of a transformer in a province of China every 15 minutes from July 2016 to July 2018. `WTH` dataset consists of 12 data fields, 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 `NHITS` has been very successful in predicting. Experiments on them provide a more comprehensive comparison of the proposed method's capabilities against `NHITS`.

The study uses macro accuracy, macro precision, macro recall, and macro F1 score to evaluate the models. Accordingly, during the inference phase, the model will run on all tasks to calculate the metrics of each one. Then, the average of the metrics of the tasks is calculated to obtain the final result.

B. Experiment

We compare the proposed method with `NHITS` using the above metrics and dataset. For each dataset, each attribute will be predicted for future trends based on past data. The final result is calculated by averaging the metrics across the attributes. We also test different feature extractors (`LSTM`, `CNN`) for the proposed method to demonstrate the deep data understanding capabilities of `LSTM+CNN`. The overall data is structured into training sets, validation sets, and testing sets with a ratio of 6:2:2 for training, fine-tuning and testing model, respectively.

For `NHITS` model, we split the data as usual according to the above predetermined ratio. For the proposed algorithm, because the meta-training and meta-testing processes require dividing the data into small tasks, and allowing the model to adapt on the support set of each task, we split the data into 40-60 tasks (see table I for statistical detail). In each task, the support set accounts for 20% with the purpose of letting the model adapt to the data, the query set accounts for 80% to check the model's compatibility. We use 50% of tasks for meta-training,

TABLE I: Statistics on USD/JPY and multi-fx.

Dataset	#task	#samples	#samples/task			
			min	mean	std	max
`USD/JPY`	60	150,175	2,502	2,502.92	0.28	2,503
`multi-fx`	60	154,440	2,467	2,575	58.61	2748
`ETT-m2`	48	69,680	1,451	1,451.67	0.47	1,452
`WTH`	40	35,064	876	876.6	0.49	877

25% for meta-validation, and 25% for meta-testing. With this division, we ensure the fairness that the ML model is trained with the same amount of data as `NHITS`. After the data configuration, we perform experiment as in appendix A and B.

V. RESULT & DISCUSSION

Table II compares the results between the proposed model and the `NHITS` model on the above datasets. On aperiodic data, our method achieves a much higher convergence rate than the `NHITS` model on all metrics. Indeed, the accuracy improves from 5% to 11% compared to `NHITS` on `USD/JPY` and `multi-fx` data. The remaining metrics also increase from 5% to 20%. On periodic data (`ETT-m2` and `WTH` dataset), our method obtains results that are higher than or close to those of `NHITS`. Specifically, our results differ by 1-3% on all metrics compared to the results of `NHITS`. In the following subsections, we analyze these results on aperiodic datasets based on two factors: (1) - Feature extraction ability; (2) - Model synthesis ability; and relate to periodic datasets.

A. Feature extraction ability

We trained three models `CNN`, `LSTM`, `LSTM+CNN` on 2600 samples of `USD/JPY` data in the conventional way. After observing their training process (figure 3), we found that `CNN` has excellent local feature extraction ability on short-term data. On the other hand, `LSTM` does not improve after 40 epochs, indicating that focusing on exploiting short/long-term dependencies without extracting deep enough features will not yield good results. When combining both features, the model shows an improvement in learning ability on the training set. The convergence rate is generally higher than when using each model separately. This can be explained by the fact that `LSTM` successfully integrated long-term features into the model, as well as taking advantage of the local features of `CNN`.

However, when observing the results on the validation set, all three models above give very poor results. To explain this, we think that although they can learn well on the training set, the models lose their generalization ability because this is a complex type of data, different from data such as images, videos, or periodic data. If

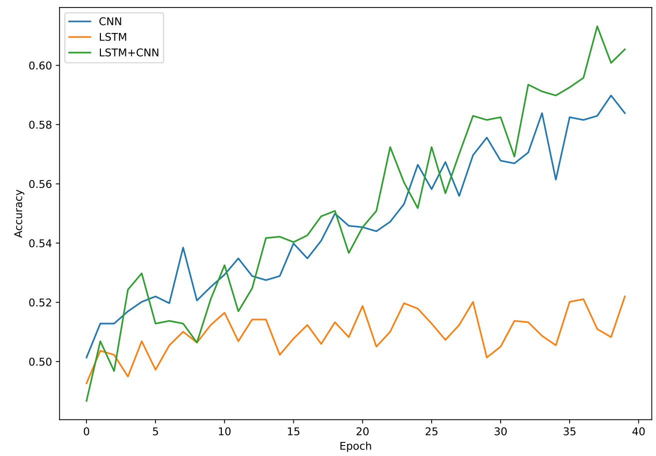

FIG. 3: Training process of LSTM, CNN, and LSTM+CNN using 2600 samples from USD/JPY dataset.

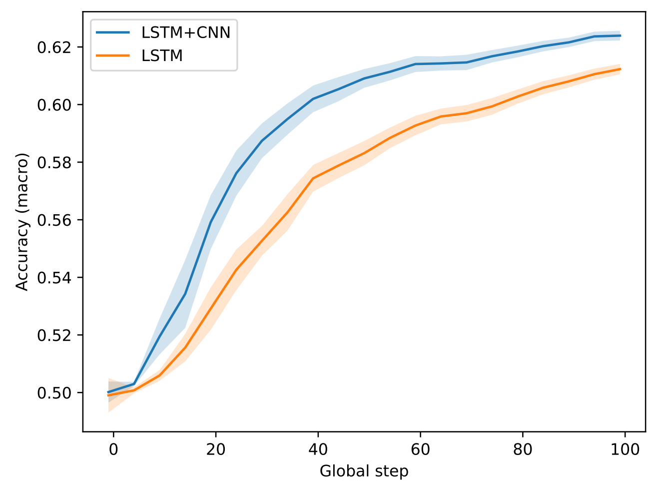

FIG. 4: Convergence process on validation set of ML model using LSTM+CNN feature and LSTM feature only (predict close price of multi-fx dataset). The blur domains cover 99.73% ($\pm 3\sigma$) values of accuracies.

the model's adaptability is not high, poor results on the validation set are predictable. However, **the graph in figure 3 shows that we cannot deny the feature extraction ability of `LSTM+CNN`.**

The feature extraction process of `NHITS` shows that this method is trying to simulate the frequency resolution process of Fourier transform. The frequency 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 resolve the frequency on non-periodic data. Moreover, it is not difficult to see that the feature extraction techniques of `NHITS` are extremely simple (only including `FullyConnected` and `Pooling` classes). These are the reasons why `NHITS` encountered great difficulties

TABLE II: Classification results (%) of NHITS and our method using USD/JPY and multi-fx datasets. Best results per metrics are boldfaced.

		accuracy	*precision*	*recall*	*F1 − score*
USD/JPY	NHITS	58.09	58.32	57.53	56.53
	Ours(LSTM)	69.45 ± 1.2	69.87 ± 1.04	69.49 ± 0.59	68.89 ± 1.43
	Ours(LSTM+CNN)	$\mathbf{69.67 \pm 1.07}$	$\mathbf{70.12 \pm 1.02}$	$\mathbf{69.74 \pm 0.33}$	$\mathbf{69.08 \pm 0.34}$
multi-fx	NHITS	52.63	52.62	52.59	52.49
	Ours(LSTM)	63.63 ± 2.79	64.58 ± 2.65	63.87 ± 1.93	62.33 ± 5.07
	Ours(LSTM+CNN)	$\mathbf{63.78 \pm 2.4}$	$\mathbf{64.84 \pm 0.76}$	$\mathbf{64.06 \pm 1.67}$	$\mathbf{62.37 \pm 3.89}$
ETT-m2	NHITS	71.72	**67.22**	**63.69**	**63.28**
	Ours(LSTM)	71.41 ± 6.36	63.48 ± 2.41	60.57 ± 3.16	59.77 ± 3.42
	Ours(LSTM+CNN)	$\mathbf{72.09 \pm 5.37}$	64.12 ± 2.66	61.13 ± 2.78	60.45 ± 3.19
WTH	NHITS	74.17	68.18	66.77	67.08
	Ours(LSTM)	74.65 ± 2.31	68.76 ± 2.38	65.68 ± 1.8	65.84 ± 2.03
	Ours(LSTM+CNN)	$\mathbf{75.45 \pm 2.07}$	$\mathbf{69.64 \pm 1.96}$	$\mathbf{67.18 \pm 1.49}$	$\mathbf{67.09 \pm 1.82}$

on multi-fx, USD/JPY dataset and non-periodic data in general. On the other hand, our proposed method uses deep learning networks for feature extraction, so it can easily obtain deep hidden features. Therefore, on both periodic and aperiodic data, our method achieves results approximately equal to or better than NHITS.

B. Sub-model synthesis ability

The reason why conventional training methods do not perform well despite the improved feature extraction capabilities is due to the lack of the ability to use features effectively. This ability in previous studies has often been improved by using ensemble models. We tried using traditional models (boosting, stacking) to increase the model accuracy but soon realized that this method was not feasible. Even when we performed brute-force search the hyper-parameter space to find the best sub-model architecture using AutoKeras [18], the accuracy on USD/JPY was only 52.53% and 53.71%, respectively.

When using MAML to synthesize the features of the LSTM and LSTM+CNN models, the compatibility (indicated by early convergence) as well as the results (indicated by accuracy) of the model using LSTM+CNN are significantly higher than those of the model using only LSTM (see figure 4) and NHITS model. Indeed, on multi-fx dataset, after only 40 epochs, the LSTM+CNN features converge to over 60% and after 100 epochs, the model converges to 62%, exceeding what the LSTM features can do. The variation of the accuracy values is very small, indicating the stability of MAML. For the ML model using only CNN, the accuracy and error values do not even improve during the entire training process. We explain this by saying that the features of CNN focus too much on the locality of the data and ignore the long-term dependencies, which is an important feature of time-series data. In contrast, the features of LSTM+CNN not only capture the long-term dependencies but also highlight the local properties in the data, making it completely outperform CNN as well as NHITS. Therefore, **we conclude that ML is more flexible and efficient in synthesizing sub-models than traditional rigid ensemble models**.

When approaching USD/JPY, ETT-m2, and WTH dataset using ML, the model also shows accuracies higher than or close to NHITS. By looking at the past time periods, **the proposed method has proven to be more effective in capturing hidden long-term dependencies through the efficient aggregation of features that the LSTM easily forgets during training**. This proves the hypothesis of the existence of hidden long-term dependencies that we stated in section I.

In the process of synthesizing features to make the final prediction, NHITS only uses simple addition and subtraction. This is reasonable in the context of synthesizing information based on interpolation functions and input residuals and trying to simulate the synthesis of frequency bands. However, facing complexity in data requires a more complex structure to be able to synthesize features in a more reasonable way. Using interpolation on the data can be considered very simple compared to the meta-optimization process of ML algorithms. Furthermore, NHITS's separation and combination of data frequency bands can only work well if the data is truly periodic. Non-periodic data such as FX or stock prices can be a fatal weakness for this approach.

VI. CONCLUSION & FUTURE DIRECTION

Problems on aperiodic time-series data pose a great challenge to current machine learning models due to the uncertainty in variance as well as the aperiodicity of this type of data. By combining local features and long-term dependencies as well as exploiting hidden dependencies in each past time period, we have improved the ability of machine learning models on classification metrics in the problem of predicting the next day's price trend on FX data. The proposed algorithm has demonstrated its superior performance when compared to `NHITS` on aperiodic data as well as achieving an accuracy close to `NHITS` on periodic data.

In the future, we propose two main directions of development related to the architecture and personalization of the learning model.

Model architecture. Our method is modularized with two main modules operating in parallel: Feature Extraction Module and Model Synthesis Module. This provides our method with flexible scalability. Our experiment only illustrates a typical case of efficient feature extraction and synthesis. By replacing different feature extraction models and using different ML algorithms (`Meta-SGD` [23], `Reptile` [27], `iMAML` [28]), it is possible to create new models with higher accuracy.

Long-horizon problem. 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.

VII. ACKNOWLEDGEMENTS

VIII. AUTHOR CONTRIBUTIONS

All authors contributed to conceiving the idea. Bao-Long Nguyen, Tom Ichibha performed calculations. All authors contributed to the discussion and writing of the paper.

IX. DATA AVAILABILITY STATEMENT

The datasets used and/or analyzed during the current study available from the corresponding authors on reasonable request.

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

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

[3] Ayitey Junior, M., Appiahene, P., Appiah, O., Bombie, C.N.: Forex market forecasting using machine learning: Systematic literature review and meta-analysis. Journal of Big Data **10**(1), 9 (2023)

[4] Challu, C., Olivares, K.G., Oreshkin, B.N., Ramirez, F.G., Canseco, M.M., Dubrawski, A.: Nhits: Neural hierarchical interpolation for time series forecasting. In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 6989–6997 (2023)

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

[6] Cheng, L.C., Huang, Y.H., Wu, M.E.: Applied attention-based lstm neural networks in stock prediction. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 4716–4718. IEEE (2018)

[7] Dua, S., Kumar, S.S., Albagory, Y., Ramalingam, R., Dumka, A., Singh, R., Rashid, M., Gehlot, A., Alshamrani, S.S., AlGhamdi, A.S.: Developing a speech recognition system for recognizing tonal speech signals using a convolutional neural network. Applied Sciences **12**(12), 6223 (2022)

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

[9] Fama, E.F.: Efficient capital markets: A review of theory and empirical work. Journal of finance **25**(2), 383–417 (1970)

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

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

[12] He, T., Droppo, J.: Exploiting lstm structure in deep neural networks for speech recognition. In: 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). pp. 5445–5449. IEEE (2016)

[13] Heryadi, Y., Wibowo, A., et al.: Foreign exchange prediction using machine learning approach: A pilot study. In: 2021 4th International Conference on Information and Communications Technology (ICOIACT). pp. 239–242. IEEE (2021)

[14] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[15] Hospedales, T., Antoniou, A., Micaelli, P., Storkey, A.: Meta-learning in neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence **44**(9), 5149–5169 (2021)

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

[17] Islam, M.S., Hossain, E.: Foreign exchange currency rate prediction using a gru-lstm hybrid network. Soft Computing Letters **3**, 100009 (2021)

[18] Jin, H., Chollet, F., Song, Q., Hu, X.: Autokeras: An automl library for deep learning. Journal of machine Learning research **24**(6), 1–6 (2023)

[19] Khoee, A.G., Yu, Y., Feldt, R.: Domain generalization through meta-learning: A survey. arXiv preprint arXiv:2404.02785 (2024)

[20] Kolle, O.: Weather dataset (2008), `https://www.bgc-jena.mpg.de/wetter/`
[21] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems **2** (1989)
[22] Li, C., Song, D., Tao, D.: Multi-task recurrent neural networks and higher-order markov random fields for stock price movement prediction: Multi-task rnn and higer-order mrfs for stock price classification. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1141–1151 (2019)
[23] Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835 (2017)
[24] Mech, T.S.: Portfolio return autocorrelation. Journal of Financial Economics **34**(3), 307–344 (1993)
[25] Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R.J., Fredes, C., Valenzuela, A.: A review of convolutional neural network applied to fruit image processing. Applied Sciences **10**(10), 3443 (2020)
[26] Nguyen, B.L., Cao, T.C., Le, B.: Meta-learning and personalization layer in federated learning. In: Asian Conference on Intelligent Information and Database Systems. pp. 209–221. Springer (2022)
[27] Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
[28] Rajeswaran, A., Finn, C., Kakade, S.M., Levine, S.: Meta-learning with implicit gradients. Advances in neural information processing systems **32** (2019)
[29] Sadeghi, A., Daneshvar, A., Zaj, M.M.: Combined ensemble multi-class svm and fuzzy nsga-ii for trend forecasting and trading in forex markets. Expert Systems with Applications **185**, 115566 (2021)
[30] Sharma, N., Jain, V., Mishra, A.: An analysis of convolutional neural networks for image classification. Procedia computer science **132**, 377–384 (2018)
[31] Varshitha, K.S., Kumari, C.G., Hasvitha, M., Fiza, S., Amarendra, K., Rachapudi, V.: Natural language processing using convolutional neural network. In: 2023 7th International Conference on Computing Methodologies and Communication (ICCMC). pp. 362–367. IEEE (2023)
[32] Vettoruzzo, A., Bouguelia, M.R., Vanschoren, J., Rognvaldsson, T., Santosh, K.: Advances and challenges in meta-learning: A technical review. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
[33] Vo, Q.H., Nguyen, H.T., Le, B., Nguyen, M.L.: Multi-channel lstm-cnn model for vietnamese sentiment analysis. In: 2017 9th international conference on knowledge and systems engineering (KSE). pp. 24–29. IEEE (2017)
[34] Zafeiriou, T., Kalles, D.: 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 (2020)
[35] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 11106–11115 (2021)

Appendix A: Experimental detail for proposed method

We use a `FullyConnected` layer of 16 units with a `ReLU` activation function to decompose the initial feature. This feature is then passed in parallel to the `BidirectionalLSTM` and `CNN` blocks. The `BidirectionalLSTM` block consists of 32 hidden units, the outputs of which are concatenated to form a final vector. The `CNN` block consists of two `CNN` layers with 32 and 64 filters, respectively. The kernel used in the layers is of size 3×3. Each `CNN` layer is followed by a `MaxPooling` layer using a kernel of size 2×2. The `CNN` block ends with a `Flatten` layer. Features of the `BidirectionalLSTM` and `CNN` blocks are then concatenated and 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 steps, outer training steps. To facilitate fine-tuning, we fix most of the parameters and only fine-tune the size of lookback window, the inner and outer learning rates. Details are presented in table III.

TABLE III: 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, 0.005, 0.01, 0.05}
Outer learning rate	{0.001, 0.005, 0.0015, 0.0055}

Appendix B: Experimental detail for NHITS

We rely on [4] to define the search space (table IV) for parameter fine-tuning, as well as to fine-tune the model architecture. For parameters not mentioned in the table, we use the default values of the `NHITS` implementation in the `NeuralForecast` library [11]. The best results of fine-tuning process are selected and reported in this study.

TABLE IV: 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	{5, 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 MLF layers	{1,2}
Learning rate	{0.001, 0.002, 0.005, 0.01, 0.02}