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

Bao-Long Nguyen,[1,*] Tom Ichibha,[1,†] Kenta Hongo,[2,‡] and Ryo Maezono[1,§]
[1]*School of Information Science, JAIST, Ishikawa, Japan*
[2]*Research Center for Advanced Computing Infrastructure, JAIST, Ishikawa, Japan*
(Dated: August 27, 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 variance and is not stationary; does not show clear cycles, 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 higher accuracy than 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 long been a matter of concern for many researchers [11, 15, 18]. The two main techniques used in aperiodic time-series prediction are fundamental analysis and technical analysis [2]. While fundamental analysis focuses on analyzing external factors, which are difficult to be captured from past value fluctuations such as policies and economic strategies of companies and countries to predict the future; technical analysis relies entirely on historical value fluctuations to analyze future trends.

Predicting on aperiodic time-series data faces several inherent challenges: (1) - The variance of this type of data varies greatly over time. Therefore, the assumption that they follow a distribution to approximate the error cannot be used, leading to machine learning models having difficulty accurately predicting future values and trends; (2) - Aperiodic data does not follow any explicit rules, so learning the hidden features of the data to make predictions is very difficult; (3) - Aperiodic time-series data (e.g. a company's stock price) depends not only on past data but also on many external factors (e.g. news, economic situation, politics) [18].

For the first challenge, ensemble models [cite something here] are often used to mitigate the effects of variance variation. Ensemble models provide a holistic, multi-perspective view based on sub-models, thereby helping the overall model adapt to strong variance changes. We approach the problem in a similar but higher-level way using Meta-learning (ML) [9]. This method effectively aggregates the parameters of local models, which helps to significantly reduce variance loss.

To overcome the second challenge, most studies use features extracted from Long short-term memory neural network (`LSTM`) [12], Artificial neural network (`ANN`), and Convolution neural network (`CNN`) [17]. Specifically, in 2022, 20% of all publications related to financial index prediction used `LSTM`, 20% used `ANN`, and 6% used `CNN` [2]. To make full use of the features extracted by the above models, we propose a method that combines these features.

Regarding the third challenge, [8] research hypothesizes that different time-series datasets of the same domain at the same time point reflect the impact of extraneous factors. For example, studies [1, 20] show the dependence between the financial ratios of a given company and the ratios of other companies. This further strengthens the hypothesis in study [8]. In addition, we argue that this type of data also has hidden long-term dependencies. For the traditional approach, people use a fixed amount of past data (lookback window) to train the model. This causes a big obstacle to the learning process because long-term features over time will be forgotten. 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 proposed algorithm by solving the problem of predicting the trend (up or down) of foreign exchange rates and comparing the results with the existing state-of-the-art (SOTA) model (`NHITS` [3]) 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 [insert github link here].

In summary, our main contributions are as follows:

- Feature combination: Extract feature using `LSTM` and `CNN` and combine them.
- Hidden long-term dependency: Experimentally

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

demonstrate that a given aperiodic time-series data not only depends on external factors but also has hidden dependencies with itself at various points in the past.

- Efficient model parameter aggregation: Use ML instead of traditional ensemble models to aggregate results from machine learning models.
- Experiment: Experiment on exchange rate datasets and compare with `NHITS` - the SOTA model to demonstrate the effectiveness of the proposed method.

II. RELATED WORK

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 [5]. In addition, `LSTM` performs feature extraction with nonlinear activation functions, which helps the model parameters capture the nonlinearity of the data [10]. 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 [21, 23] 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 [6], natural language processing [24]. 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. Model-agnostic Meta-learning (MAML)

Meta-learning (ML) algorithms, typically MAML [9] 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 [13, 25]. 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 [4, 7, 22], domain adaptation in online learning [14, 16]).

A basic ML algorithm is trained on multiple tasks t drawn from the same task distribution $\mathcal{T}$ [13]. 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 [2]. However, current ensemble model synthesis forms are still very rigid because they can only synthesize based on the final results (voting mechanism of bagging models) and near-final results (for 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 [3], 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 [3]):

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

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

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

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

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

Accordingly, `FullyConnected` are stacked multi-layer perception (`MLP`) 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 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`.

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.

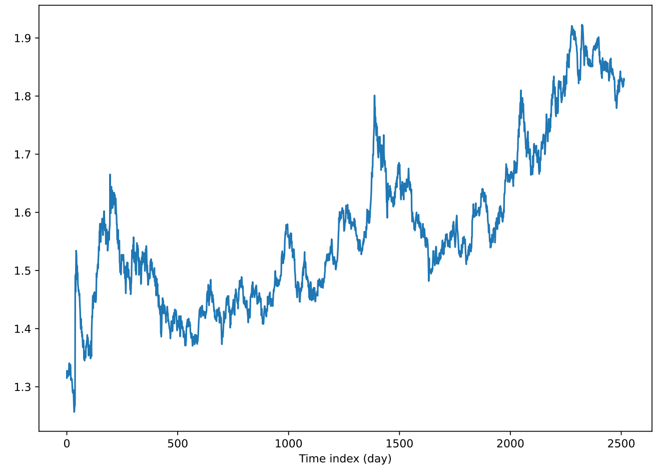

FIG. 1. Exchange rate (close price) between Swiss franc and New Zealand dollar by day (2014-2024).

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.

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: $\mathcal{D} = \{\mathcal{D}_t\}^n_{t=1}$. The data at each task is divided into support and query sets: $\mathcal{D}_t = \left\{\mathcal{D}^{support}_t, \mathcal{D}^{query}_t\right\}$.

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.

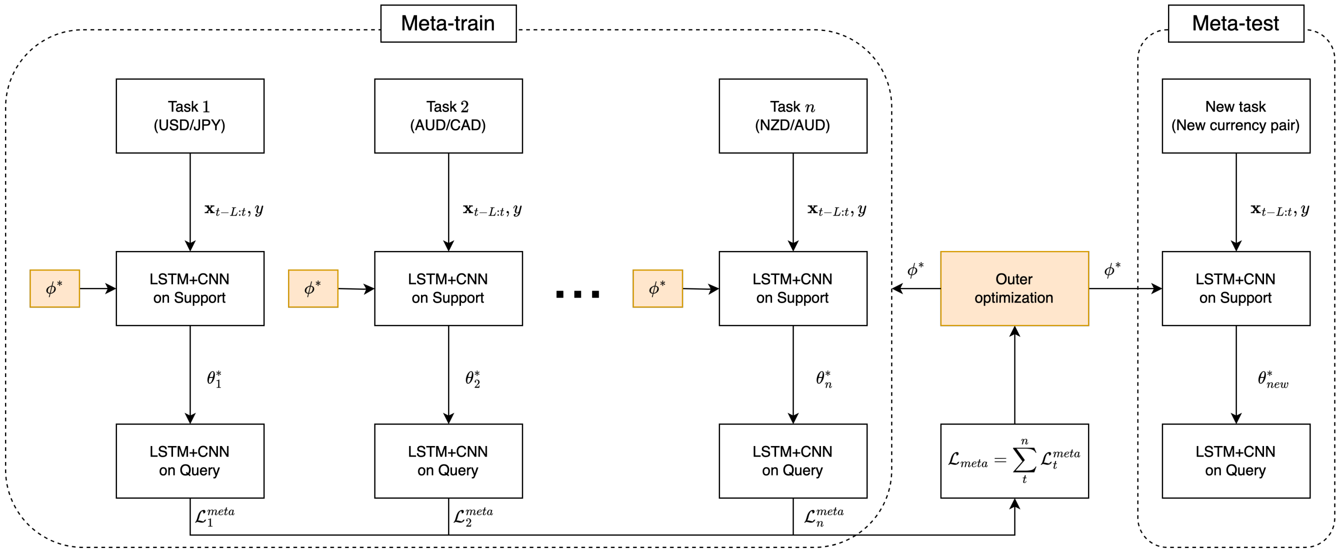

FIG. 2. The full-flow of meta-training and meta-testing on multi-fx data. Each currency pair is regarded as a task.

B. Feature extraction

Inspired by the [26] 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 a MLP layer 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:

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

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).

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

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, the eth local optimization process at the support set of task t occurs as follows:

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

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).

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

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):

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

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 [4, 7, 9, 19, 22] 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. The `USD/JPY` dataset consists of only data of the USD/JPY currency pair, 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. The `multi-fx` dataset consists of 60 currency pairs made up 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. The number of data samples of these two datasets are similar and approximately 156000 samples.

The `multi-fx` dataset is used 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 [1, 8, 20]. The `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.

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 each task to calculate the metrics of each task. 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. The overall data is structured into training sets, validation sets, and testing sets with a ratio of 6:2:2. The training set is used to train the model, the validation set is used to find hyper-parameters, and the test set is used to evaluate the model. For `NHITS`, 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 60 tasks. 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 30 tasks for meta-training, 15 tasks for meta-validation, and 15 tasks for meta-testing. With this division, we ensure the fairness that the ML model is trained with the same amount of data as `NHITS`.

In our implementation, 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.

V. RESULTS AND DISCUSSION

VI. CONCLUSION & FUTURE DIRECTION

VII. ACKNOWLEDGEMENTS

VIII. AUTHOR CONTRIBUTIONS

IX. DATA AVAILABILITY STATEMENT

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

[2] 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.

[3] 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.

[4] 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 .

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

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

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

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

[9] 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.

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

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

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

[13] 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.

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

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

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

[17] 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.

[18] Li, C., Song, D., Tao, D., 2019. 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.

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

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

[21] Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R.J., Fredes, C., Valenzuela, A., 2020. A review of convolutional neural network applied to fruit image processing. Applied Sciences 10, 3443.

[22] 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.

[23] Sharma, N., Jain, V., Mishra, A., 2018. An analysis of convolutional neural networks for image classification. Procedia computer science 132, 377–384.

[24] Varshitha, K.S., Kumari, C.G., Hasvitha, M., Fiza, S., Amarendra, K., Rachapudi, V., 2023. Natural language processing using convolutional neural network, in: 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), IEEE. pp. 362–367.

[25] 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 .

[26] Vo, Q.H., Nguyen, H.T., Le, B., Nguyen, M.L., 2017. Multi-channel lstm-cnn model for vietnamese sentiment analysis, in: 2017 9th international conference on knowledge and systems engineering (KSE), IEEE. pp. 24–29.

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

		accuracy	*precision*	*recall*	*F*1
`USD/JPY`	`NHITS`	53.29	52.38	52.23	51.87
	Ours	$\mathbf{58.05 \pm 0.01}$	$\mathbf{59.57 \pm 2.53}$	$\mathbf{58.31 \pm 2.57}$	$\mathbf{56.6 \pm 3.74}$
`multi-fx`	`NHITS`	51.12	51.22	51.16	50.54
	Ours	$\mathbf{62.39 \pm 0.06}$	$\mathbf{71.11 \pm 9.34}$	$\mathbf{70.14 \pm 9.6}$	$\mathbf{69.21 \pm 11.07}$