HyperNetwork Approximating Future Parameters for Time Series Forecasting under Temporal Drifts

Jaehoon Lee$_{1,2}$[*]**, Chan Kim$_1$, Gyumin Lee$_1$, Haksoo Lim$_1$, Jeongwhan Choi$_1$, Kookjin Lee$_3$, Dongeun Lee$_4$, Sanghyun Hong$_5$, and Noseong Park$_1$**
Yonsei University$_1$, LG AI Research$_2$, Arizona State University$_3$,
Texas A&M University-Commerce$_4$, Oregon State University$_5$

Abstract

Models for time series forecasting require the ability to extrapolate from previous observations. Yet, extrapolation is challenging, especially when the data spanning several periods is under *temporal drifts* where each period has a different distribution. To address this problem, we propose HyperGPA, a hypernetwork that generates a target model's parameters that are expected to work well (i.e., be an optimal model) for each period. HyperGPA discovers an underlying hidden dynamics which causes temporal drifts over time, and generates the model parameters for a target period, aided by the structures of computational graphs. In comprehensive evaluations, we show that target models whose parameters are generated by HyperGPA are up to 64.1% more accurate than baselines.

1 Introduction

Time series forecasting is one of the most fundamental problems in deep learning, ranging from classical climate modeling [Brouwer et al., 2019, REN, 2021] and stock price forecasting [Ariyo et al., 2014, Vijh et al., 2020] to pandemic forecasting [Wu et al., 2020, Wang et al., 2021a]. Owing to recent novel methods, the forecasting accuracy has been significantly enhanced over the past several years [Kidger et al., 2020, Zhou et al., 2021]. However, this forecasting is challenging, especially when *temporal drifts*, i.e., a data distribution changes over time by an underlying latent dynamics, exist in time series data [Zhang et al., 2013, Oh et al., 2019, Li et al., 2021, Kuznetsov and Mohri, 2014, Kuznetsov et al., 2015]. For example, the number of COVID-19 patients fluctuates severely over time, and a dynamics behind the daily patient numbers is governed by complicated factors.

To defend against the latent dynamics, we present a **Hyper**network **G**enerating **P**arameters in **A**dvance (**HyperGPA**). In usual settings, to forecast the future, one can train the parameters of time series models with data from all historical periods. In contrast, *HyperGPA* reads some recent periods and forecasts the parameters of other neural networks (called *target models*) that are expected to work well in the future (cf. Fig. 1).

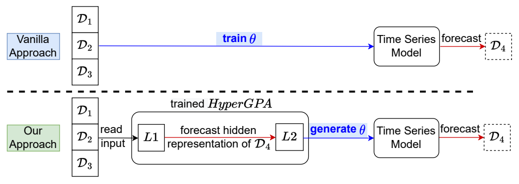


Figure 1: Comparison between vanilla and our approaches. $\mathcal{D}_j$ means the j-th period.

Our hypernetwork has two parts ($L1$ and $L2$). Firstly, $L1$ is responsible for discovering a hidden underlying dynamics and forecasting a future period's characteristic from recent periods. By integrating recent graph neural networks (GNNs) and neural controlled differential equations (NCDEs) [Kidger et al., 2020] into a single framework in $L1$, we construct a more general method which can process regular and irregular time series, and M coupled time series simultaneously. As for $L2$, it generates

[*]The work is done when the author was in Yonsei University.

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

the future parameters of target models from the hidden representation of the future period forecast by $L1$. The parameters are generated based on GNNs since target neural network models are typically represented by computation graphs.

In our experiments, we test *HyperGPA* on 24 experimental settings constructed with 4 real-world datasets and 6 target models. *HyperGPA* achieves the best forecasting performance against 6 baselines. Futhermore, it can be successfully applied to various target models for time series forecasting, ranging from RNNs and NODEs to NCDEs. To sum up, we make the following contributions:

1. We propose *HyperGPA* which forecasts the future parameters of target models from previous periods. Aided by GNNs and NCDEs, *HyperGPA* can be extended to more general cases.
2. When generating the parameters, additional GNNs are employed for the computational graph of the parameters in target models.
3. *HyperGPA* improves target models' errors up to 64.1% against several baseline methods.

2 Related works

When source and target distributions are different, domain adaptation or generalization [Ben-David et al., 2007, Wang et al., 2021b] can be a solution. However, these methods cannot be used for time series under temporal drifts, because the distribution (domain) of this data continuously changes. Therefore, different strategies are needed to handle temporal drifts. *AdaRNN* focuses on commonalities across all periods [Du et al., 2021] for addressing temporal drifts. Also, *RevIN* is an normalization layer which removes varying statistical properties from input [Kim et al., 2022]. In spite of their success under temporal drifts, their common limitation is that they learn a general time series model applicable to all periods. Our *HyperGPA* has fundamentally different approaches in that it aims to generate the optimal parameters of target models per period. Although Bai et al. [2023] is similar to our method, we have advantages in that *HyperGPA* predicts future parameters based on past data whereas Bai et al. [2023] utilized past parameters which might contain a larger variance than data.

3 Methods

3.1 Problem definition & notations

Let $\mathcal{D}_{i,j} = \{\mathbf{x}_{i,j}^k\}_{k=1}^{|\mathcal{D}_{i,j}|}$ be the i-th time series at the j-th (disjoint) period, where $\mathbf{x}_{i,j}^k \in \mathbb{R}^{\dim(\mathbf{x})}$ denotes the k-th observation. For example, in the U.S. flu dataset, $\mathcal{D}_{i,j}$ denotes time series of the i-th state at the j-th year. Then, our task is to train a hypernetwork to i) understand coupled underlying hidden dynamics causing the drifts, ii) predict the characteristics of $\{\mathcal{D}_{i,N}\}_{i=1}^M$ (as a form of latent vectors $\{\mathbf{h}_{i,N}\}_{i=1}^M$), and iii) generate the parameters of target models, which works well for $\mathcal{D}_{i,N}$. We represent the target model of the i-th time series as $\mathcal{B}_i$. $\hat{\boldsymbol{\theta}}_{i,j}$ denotes the parameters of $\mathcal{B}_i$ which are generated by *HyperGPA* and expected to work well in the j-th period. K is an input period size (cf. Fig. 2). The task of $\mathcal{B}_i$ is to forecast the next s_{out} observations given past s_{in} observations. We represent the adjacency matrix of the computation graph of $\mathcal{B}_i$ as $\mathbf{A}$. We assume that all $\mathcal{B}_i$ have the same structure $\mathbf{A}$. v_l is the l-th vertex of $\mathbf{A}$, which denotes the l-th parameter of $\mathcal{B}_i$ (e.g., a weight or a bias). L is the number of vertices in $\mathbf{A}$.

3.2 Architecture of HyperGPA

Let $\mathcal{D}_{input} = \{\{\mathcal{D}_{i,j}\}_{j=N-K}^{N-1}\}_{i=1}^M$ be the collected time series data for K recent periods Then, $L1$ reads the previous periods $\mathcal{D}_{input}$ and forecasts $\{\mathbf{h}_{i,N}\}_{i=1}^M$, the hidden representation of the next period $\{\mathcal{D}_{i,N}\}_{i=1}^M$. At this step, we utilize the GNNs to discover relationships among those M different correlated time series. As for GNNs, we employ *adaptive graph convolutional* (AGC) function to discover *latent* relationships. Therefore, our method can be used even when the explicit relationships among M time series are unknown, which improves the applicability. We use the following method to combine NCDEs and AGC into a single framework:

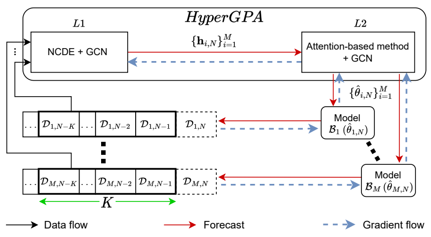


Figure 2: Overall workflow of *HyperGPA*

$$\mathbf{h'}_i(1) = \Gamma_i(\mathbf{x}_i(1); \boldsymbol{\theta}_{\Gamma_i}), \tag{1}$$

$$\mathbf{h'}_i(T) = \mathbf{h'}_i(1) + \int_1^T G(\{\mathbf{h'}_i(t)\}_{i=1}^M; \boldsymbol{\theta}_G) \mathrm{d}\mathbf{X}_i(t), \tag{2}$$

where $\mathbf{h}_{i,N} = \mathbf{h'}_i(T)$ and $\mathbf{h'}_i(t)$ is an NCDE hidden vector of the i-th time series at integral time t. Also, the final integral time T is the total time length of input for recent periods, $T = \sum_{j=N-K}^{N-1} |\mathcal{D}_{i,j}|$. T should be the same for all i-th time series because they should evolve together with a graph function in NCDEs. $\{\mathbf{x}_i(1)\}_{i=1}^M$ are feature vectors at initial time in $\mathcal{D}_{input}$, and $\{\mathbf{X}_i(t)\}_{i=1}^M$ are interpolated lines of $\mathcal{D}_{input}$. Each Γ_i maps the initial feature vector $\mathbf{x}_i(1)$ into the initial NCDE hidden vector $\mathbf{h'}_i(1)$. Along the interpolated path, NCDEs generate the final NCDE hidden vector, $\mathbf{h'}_i(T)$. Because G incorporates the AGC function, all i-th time series are processed simultaneously through NCDEs and help each other to forecast the next hidden representation.

Given $\mathbf{h}_{i,N}$, we use an attention-based parameter generation method and GCNs to generate $\hat{\boldsymbol{\theta}}_{i,N}$:

$$\{\mathbf{z}_{i,N}^l\}_{l=1}^L = \Phi(\mathbf{h}_{i,N}; \boldsymbol{\theta}_\Phi), \tag{3}$$

$$\{\mathbf{q}_{i,N}^l\}_{l=1}^L = \mathtt{GNN}(\{\mathbf{z}_{i,N}\}_{l=1}^L; \mathbf{A}, \boldsymbol{\theta}_{\mathtt{GNN}}), \tag{4}$$

$$\hat{\boldsymbol{\theta}}_{i,N}^l = \mathtt{Attn}(\mathbf{q}_{i,N}^l, \{\boldsymbol{\theta}_{\text{key},c}^l\}_{c=1}^C, \{\tilde{\boldsymbol{\theta}}_c^l\}_{c=1}^C), \tag{5}$$

where $\mathtt{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathtt{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}$, where d_k is the size of key vectors. For parameter generation, we use an attention-based method [Vaswani et al., 2017]. Query vectors $\{\mathbf{q}_{i,N}^l\}_{l=1}^L$ for a parameter vertex v_l are mapped from $\mathbf{h}_{i,N}$ by Φ and GNNs with $\mathbf{A}$. Values are candidate parameters $\{\tilde{\boldsymbol{\theta}}_c^l\}_{c=1}^C$ and each candidate has a key $\{\boldsymbol{\theta}_{\text{key},c}^l\}_{c=1}^C \in \mathbb{R}^{dim(\mathbf{q})}$, where C is the number of candidate parameters. After generating $\hat{\boldsymbol{\theta}}_{i,N}$, $\mathcal{B}_i$ equipped with it performs a seq-to-seq forecasting task in $\mathcal{D}_{i,N} = \{\mathbf{x}_{i,b}^k\}_{k=1}^{|\mathcal{D}_{i,b}|}$, reading $\{\mathbf{x}_{i,b}^k\}_{k=n-s_{in}}^{n-1}$ and forecasting $\{\mathbf{x}_{i,b}^k\}_{k=n}^{n+s_{out}-1}$.

Training procedure. Let $\{\{\mathcal{D}_{i,j}\}_{j=1}^{N-1}\}_{i=1}^M$ be the entire training time series data. Then, we create several mini-batches $(\{\{\mathcal{D}_{i,j}\}_{j=b-K}^{b-1}\}_{i=1}^M, \{\mathcal{D}_{i,b}\}_{i=1}^M)$, where $K+1 \leq b \leq N-1$. We train *HyperGPA* using mean squared error (MSE) between the forecasting results of the target model and ground truth. In a training stage, *HyperGPA* is trained by two types of MSE: one is from $\mathcal{B}_i$ configured with $\hat{\boldsymbol{\theta}}_{i,b}$ and the other is with $\tilde{\boldsymbol{\theta}}_{c^*}^l$ where c^* denotes an index which has the maximum attention score. With the second MSE, each candidate becomes meaningful (i.e. forecasts well).

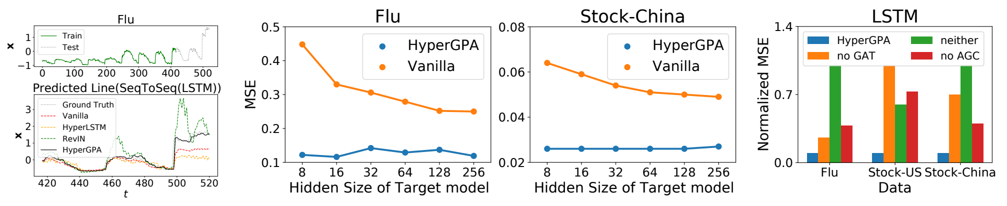
4 Experiments

4.1 Experimental setup

Datasets. We evaluate our approach on four time series forecasting benchmarks ranging from pandemic and stock to climate datasets: `Flu`, `Stock-US`, `Stock-China`, and `USHCN`. In our evaluations, we use the last and the second to the last window as a test and a validation set, respectively.

Target models & baselines. The target model is a time series model which performs a seq-to-seq forecasting task, whereas the baseline is a way to generate or train the target model's parameters. For target models, we use six time series models: i) `LSTM` [Hochreiter and Schmidhuber, 1997], ii) `GRU` [Cho et al., 2014], iii-iv) `SeqToSeq` equipped with `LSTM` and `GRU` [Sutskever et al., 2014], v) `ODERNN` [Rubanova et al., 2019], vi) `NCDE` [Kidger et al., 2020]. For baselines, there are four categories: i) In a vanilla method, one can directly train the parameters of a target model without hypernetworks or approaches addressing temporal drifts, denoted by *Vanilla*. ii) There are two hypernetwork-based methods. *HyperLSTM* or *HyperGRU* [Ha et al., 2016] are for `LSTM`, `GRU`, `SeqToSeq(LSTM)`, and `SeqToSeq(GRU)`, and *ANODEV2* [Zhang et al., 2019] is for `ODERNN` and `NCDE`. iii) The third type is approaches to address the temporal drifts. *RevIN* [Kim et al., 2022] can be used for various target models, whereas *AdaLSTM* and *AdaGRU* [Du et al., 2021] are for `LSTM` and `GRU` only, respectively. iv) Lastly, there is a statistical model, *ARIMA* [Hillmer and Tiao, 1982].

Others. For validation and test metric, we use MSE. We run 5 times and report their mean and standard deviation values. Refer to an appendix for additional information for experimental settings.



(a) Forecasting results (b) Senstivity to the hidden size of target models (c) Ablation study to GNN

Figure 3: (a) Forecasting results when data is `Flu` and a target model is `SeqToSeq (LSTM)`. (b) Sensitivity to the hidden size of the two target models, `GRU` with `Flu`, and `LSTM` with `Stock-China`. (c) Ablation with or without GNNs. To address different scale, min-max normalizations are applied to MSE scores where the min and max value is 0.1, 1, respectively.

Table 1: Test MSE in forecasting tasks. The best result for each target model is in **boldface** and for all target models with asterisk*. For *ARMIA*, we remove it to due to the worst performance.

Target Model	Method	`Flu`	`Stock-US`	`Stock-China`	`USHCN`	Target Model	Method	`Flu`	`Stock-US`	`Stock-China`	`USHCN`
`LSTM`	*Vanilla*	0.367	0.213	0.050	0.239	`GRU`	*Vanilla*	0.275	0.102	0.037	0.232
	HyperLSTM	0.582	0.751	0.103	0.249		*HyperGRU*	0.520	0.462	0.075	0.244
	RevIN	0.506	0.063	0.049	0.589		*RevIN*	0.379	0.060	0.040	0.465
	AdaLSTM	0.740	0.379	0.321	0.595		*AdaGRU*	0.616	0.233	0.170	0.472
	HyperGPA	**0.118**	**0.050**	**0.026**	**0.221**		*HyperGPA*	**0.116***	**0.052**	**0.026**	**0.229**
`SeqToSeq (LSTM)`	*Vanilla*	0.353	0.167	0.045	0.236	`SeqToSeq (GRU)`	*Vanilla*	0.250	0.112	0.035	0.232
	HyperLSTM	0.559	0.643	0.097	0.243		*HyperGRU*	0.502	0.464	0.073	0.236
	RevIN	0.345	0.061	0.044	0.585		*RevIN*	0.291	0.060	0.039	0.519
	HyperGPA	**0.128**	**0.048***	**0.026**	**0.220***		*HyperGPA*	**0.130**	**0.049**	**0.025***	**0.222**
`ODERNN`	*Vanilla*	0.361	0.200	0.056	0.235	`NCDE`	*Vanilla*	0.387	0.130	0.040	0.234
	ANODEV2	0.298	0.120	0.037	0.233		*ANODEV2*	0.821	0.244	0.141	0.483
	RevIN	0.549	0.068	0.048	0.855		*RevIN*	0.439	0.060	0.040	0.713
	HyperGPA	**0.134**	**0.050**	**0.026**	**0.226**		*HyperGPA*	**0.167**	**0.049**	**0.027**	**0.227**

4.2 Experimental results

Forecasting results. In Table 1, *HyperGPA* consistently outperforms all baselines for all target models. Also, in Fig. 3(a), *HyperGPA* captures temporal dynamics better than others. This shows that the superiority of our methods in forecasting under temporal drifts.

Sensitivity to target model size. We compare *Vanilla* and *HyperGPA*, varying the hidden size of target models — a larger hidden size leads to a larger target model size. As in Fig. 3(b), *HyperGPA* shows stable errors regardless of the hidden size whereas *Vanilla* does not show reliable forecasting accuracy when the target model size is small. This result shows that we can use small target models with *HyperGPA*, which drastically reduces the overheads for maintaining target models in practice.

Ablating GNNs. We use AGC in $L1$ (Eq. (2)) and GAT [Veličković et al., 2018] in $L2$ (Eq. (4)) for GNNs. In this paragraph, we remove AGC from $L1$ GAT from $L2$. Fig. 3(c) shows MSE for each case where the target model is `LSTM`. *HyperGPA* denotes our full model; *neither* means our model without AGC or GAT. In almost all cases, *HyperGPA* shows the lowest errors. Higher errors of *no AGC* show that considering different correlated time series simultaneously helps *HyperGPA* to better discover underlying dynamics. Also, *no GAT* has worse results, which means that the information of a computation graph is needed for generating reliable parameters.

Other experiments. Additional experimental results and visualization is available in an appendix.

5 Conclusion

We presented *HyperGPA* to address the temporal drifts by generating the future parameters of a target model. *HyperGPA* can be applied to various target models, showing the best performance in most cases. Also, we showed that *HyperGPA* allows target models to be small in real-world applications, which drastically reduces the maintenance overhead of target models. We leave testing *HyperGPA* on irregular cases and more complicated target models, such as transformers, as future works.

Acknowledgements

Noseong Park is the corresponding author. This work was supported by an IITP grant funded by the Korean government (MSIT) (No.2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University); No.2022-0-00113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework).

References

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *NeurIPS*, 2019.

Deep learning-based weather prediction: A survey. *Big Data Research*, 23:100178, 2021. ISSN 2214-5796. doi: https://doi.org/10.1016/j.bdr.2020.100178.

Adebiyi A. Ariyo, Adewumi O. Adewumi, and Charles K. Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, 2014. doi: 10.1109/UKSim.2014.67.

Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. Stock closing price prediction using machine learning techniques. *Procedia Computer Science*, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.03.326.

Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case, 2020.

Rui Wang, Danielle Maddix, Christos Faloutsos, Yuyang Wang, and Rose Yu. Bridging physics-based and data-driven modeling for learning dynamical systems. In Ali Jadbabaie, John Lygeros, George J. Pappas, Pablo A.&nbsp;Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, Proceedings of Machine Learning Research. PMLR, 07 – 08 June 2021a.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series, 2020.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *ICML*, 2013.

Jeeheh Oh, Jiaxuan Wang, Shengpu Tang, Michael W. Sjoding, and Jenna Wiens. Relaxed parameter sharing: Effectively modeling time-varying relationships in clinical time-series. *Proceedings of Machine Learning Research*, 106, 2019.

Zijian Li, Ruichu Cai, Tom Z. J. Fu, and Kun Zhang. Transferable time-series forecasting under causal conditional shift. *CoRR*, abs/2111.03422, 2021.

Vitaly Kuznetsov and Mehryar Mohri. Generalization bounds for time series prediction with non-stationary processes. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 260–274, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11662-4.

Kuznetsov, Vitaly, Mohri, and Mehryar. Learning theory and algorithms for forecasting non-stationary time series. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. Generalizing to unseen domains: A survey on domain generalization, 2021b. URL `https://arxiv.org/abs/2103.03097`.

Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. Adarnn: Adaptive learning and forecasting of time series, 2021.

Taesung Kim, Jinhee Kim Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *Submitted to The Tenth International Conference on Learning Representations*, 2022.

Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=sWOsRj4nT1n`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL `https://arxiv.org/abs/1706.03762`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, pages 3104–3112, 2014.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent odes for irregularly-sampled time series, 2019.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.

Tianjun Zhang, Zhewei Yao, Amir Gholami, Kurt Keutzer, Joseph Gonzalez, George Biros, and Michael Mahoney. Anodev2: A coupled neural ode evolution framework, 2019.

S. C. Hillmer and G. C. Tiao. An arima-model-based approach to seasonal adjustment. *Journal of the American Statistical Association*, 77(377):63–70, 1982. doi: 10.1080/01621459.1982.10477767.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

CDC. Centers for disease control and prevention. `https://www.cdc.gov/`, 1946.

Investing.com. Stock datasets. `https://www.investing.com/`, 2007.

USHCN. US Historical Climatology Network. `https://www.ncei.noaa.gov`, 1987.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting, 2020.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022.

Supplementary Material

HyperGRU

We define *HyperGRU* since only *HyperLSTM* is defined in Ha et al. [2016]. Let $\mathbf{x}_t$ be a time series observation at time t. GRU [Cho et al., 2014] is defined as follows:

$$\mathbf{r}_t = \sigma(W^{\mathbf{r}}_{\mathbf{x}}\mathbf{x}_t + W^{\mathbf{r}}_{\mathbf{h}}\mathbf{h}_{t-1} + b^{\mathbf{r}}), \tag{6}$$
$$\mathbf{z}_t = \sigma(W^{\mathbf{z}}_{\mathbf{x}}\mathbf{x}_t + W^{\mathbf{z}}_{\mathbf{h}}\mathbf{h}_{t-1} + b^{\mathbf{z}}), \tag{7}$$
$$\mathbf{g}_t = \delta(W^{\mathbf{g}}_{\mathbf{x}}\mathbf{x}_t + \mathbf{r}_t \odot W^{\mathbf{g}}_{\mathbf{h}}\mathbf{h}_{t-1} + b^{\mathbf{g}}), \tag{8}$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{g}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}, \tag{9}$$

where σ is a sigmoid function, δ is a hyperbolic tangent function, and $\odot$ is an element-wise multiplication. Likewise, *HyperGRU* is defined as follows:

$$\hat{\mathbf{x}}_t = \mathbf{x}_t \oplus \mathbf{h}_{t-1}, \tag{10}$$
$$\hat{\mathbf{r}}_t = \sigma(LN(W^{\hat{\mathbf{r}}}_{\hat{\mathbf{x}}}\hat{\mathbf{x}}_t + W^{\hat{\mathbf{r}}}_{\hat{\mathbf{h}}}\hat{\mathbf{h}}_{t-1} + b^{\hat{\mathbf{r}}})), \tag{11}$$
$$\hat{\mathbf{z}}_t = \sigma(LN(W^{\hat{\mathbf{z}}}_{\hat{\mathbf{x}}}\hat{\mathbf{x}}_t + W^{\hat{\mathbf{z}}}_{\hat{\mathbf{h}}}\hat{\mathbf{h}}_{t-1} + b^{\hat{\mathbf{z}}})), \tag{12}$$
$$\hat{\mathbf{g}}_t = \delta(LN(W^{\hat{\mathbf{g}}}_{\hat{\mathbf{x}}}\hat{\mathbf{x}}_t + \hat{\mathbf{r}}_t \odot W^{\hat{\mathbf{g}}}_{\hat{\mathbf{h}}}\hat{\mathbf{h}}_{t-1} + b^{\hat{\mathbf{g}}})), \tag{13}$$
$$\hat{\mathbf{h}}_t = (1 - \hat{\mathbf{z}}_t) \odot \hat{\mathbf{g}}_t + \hat{\mathbf{z}}_t \odot \hat{\mathbf{h}}_{t-1}, \tag{14}$$

where $\oplus$ means a concatenation. From $\hat{\mathbf{h}}_t$, the embeddings of each gate $\mathbf{a}^{\mathbf{h},y}_t, \mathbf{a}^{\mathbf{x},y}_t, \mathbf{a}^{\mathbf{b},y}_t$ are generated, where $y \in \{\mathbf{r}, \mathbf{z}, \mathbf{g}\}$. In addition, those embeddings are transformed into $\mathbf{d}^{\mathbf{h},y}_t, \mathbf{d}^{\mathbf{x},y}_t, \mathbf{d}^{\mathbf{b},y}_t$:

$$\mathbf{a}^{\mathbf{h},y}_t = W^{\mathbf{h},y}_{\hat{\mathbf{h}}}\hat{\mathbf{h}}_{t-1} + b^{\mathbf{h},y}_{\hat{\mathbf{h}}}, \tag{15}$$
$$\mathbf{a}^{\mathbf{x},y}_t = W^{\mathbf{x},y}_{\hat{\mathbf{h}}}\hat{\mathbf{h}}_{t-1} + b^{\mathbf{x},y}_{\hat{\mathbf{h}}}, \tag{16}$$
$$\mathbf{a}^{\mathbf{b},y}_t = W^{\mathbf{b},y}_{\hat{\mathbf{h}}}\hat{\mathbf{h}}_{t-1} + b^{\mathbf{b},y}_{\hat{\mathbf{h}}}, \tag{17}$$
$$\mathbf{d}^{\mathbf{h},y}_t = W^{\mathbf{h},y}_{\mathbf{a}}\mathbf{a}^{\mathbf{h},y}_t + b^{\mathbf{h},y}_{\mathbf{a}}, \tag{18}$$
$$\mathbf{d}^{\mathbf{x},y}_t = W^{\mathbf{x},y}_{\mathbf{a}}\mathbf{a}^{\mathbf{x},y}_t + b^{\mathbf{x},y}_{\mathbf{a}}, \tag{19}$$
$$\mathbf{d}^{\mathbf{b},y}_t = W^{\mathbf{b},y}_{\mathbf{a}}\mathbf{a}^{\mathbf{b},y}_t + b^{\mathbf{b},y}_{\mathbf{a}}. \tag{20}$$

Finally, $\mathbf{d}^{\mathbf{h},y}_t, \mathbf{d}^{\mathbf{x},y}_t, \mathbf{d}^{\mathbf{b},y}_t$ adjust the parameters of the main GRU and h_t is generated:

$$\mathbf{r}_t = \sigma(LN(\mathbf{d}^{\mathbf{x},\mathbf{r}}_t \odot W^{\mathbf{r}}_{\mathbf{x}}\mathbf{x}_t + \mathbf{d}^{\mathbf{h},\mathbf{r}}_t \odot W^{\mathbf{r}}_{\mathbf{h}}\mathbf{h}_{t-1} + \mathbf{d}^{\mathbf{b},\mathbf{r}}_t)), \tag{21}$$
$$\mathbf{z}_t = \sigma(LN(\mathbf{d}^{\mathbf{x},\mathbf{z}}_t \odot W^{\mathbf{z}}_{\mathbf{x}}\mathbf{x}_t + \mathbf{d}^{\mathbf{h},\mathbf{z}}_t \odot W^{\mathbf{z}}_{\mathbf{h}}\mathbf{h}_{t-1} + \mathbf{d}^{\mathbf{b},\mathbf{z}}_t)), \tag{22}$$
$$\mathbf{g}_t = \delta(LN(\mathbf{d}^{\mathbf{x},\mathbf{g}}_t \odot W^{\mathbf{g}}_{\mathbf{x}}\mathbf{x}_t + \mathbf{r}_t \odot \mathbf{d}^{\mathbf{h},\mathbf{g}}_t \odot W^{\mathbf{g}}_{\mathbf{h}}\mathbf{h}_{t-1} + \mathbf{d}^{\mathbf{b},\mathbf{g}}_t)), \tag{23}$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{g}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}. \tag{24}$$

Reproducibility

We implement *HyperGPA* with Python v3.7 and PyTorch v1.8 and run our experiments on a machine equipped with Nvidia RTX Titan or 3090. The code is available in `https://github.com/leejaehoon2016/HyperGPA/tree/main`.

Datasets. `Flu` contains the number of flu patients, total patients, and providers who give information about patients, in each of the 51 U.S. states, collected weekly by the Centers for Disease Control and Prevention [CDC, 1946], i.e., $\dim(\mathbf{x}) = 3$. The data collection period is 2011–2020. In *HyperGPA*, we set the length of one window $\mathcal{D}_{i,j}$ to a year; the length is 52 (or 53) weeks. For `Flu`, each target time series model $\mathcal{B}_i$ reads recent $s_{in} = 10$ observations and forecasts next $s_{out} = 2$ observations.

We also use the daily historical stock prices of the U.S. and China (`Stock-US` and `Stock-China`) collected by Investing.com [2007] over 20 months. Each dataset contains the opening, closing, highest, and lowest stock prices per day, i.e., $\dim(\mathbf{x}) = 4$. We choose the top-30 companies with the highest market capitalization. For those two datasets, we set the window length to two months. Note

Table 2: Hyperparameters for *HyperGPA*

Data	Target Model	$\dim(\mathbf{h}')$	$\dim(\mathbf{z})$	C	λ
Flu	LSTM	128	2048	3	0.1
	GRU	128	1024	3	0.01
	SeqToSeq(LSTM)	128	1024	10	0.1
	SeqToSeq(GRU)	128	2048	3	0.01
	ODERNN	128	2048	10	0.0
	NCDE	128	512	3	0.1
Stock-US	LSTM	128	512	3	0.0
	GRU	128	512	10	0.01
	SeqToSeq(LSTM)	128	512	3	0.1
	SeqToSeq(GRU)	128	1024	5	0.0
	ODERNN	128	512	10	0.01
	NCDE	128	512	10	0.0
Stock-China	LSTM	128	512	3	0.1
	GRU	128	512	10	0.0
	SeqToSeq(LSTM)	128	2048	5	0.1
	SeqToSeq(GRU)	128	512	3	0.01
	ODERNN	128	1024	3	0.0
	NCDE	128	1024	5	0.0
USHCN	LSTM	128	512	48	0.01
	GRU	32	512	48	0.1
	SeqToSeq(LSTM)	32	512	48	0.01
	SeqToSeq(GRU)	128	512	48	0.1
	ODERNN	64	512	48	0.1
	NCDE	32	512	48	0.01

Table 3: Hyperparameters for *Vanilla*

Data	Target Model	$\dim(\mathbf{h}_{\mathcal{B}})$	$n_{\mathcal{B}}$
Flu	LSTM	64	1
	GRU	64	1
	SeqToSeq (LSTM)	64	1
	SeqToSeq (GRU)	64	1
	ODERNN	32	1
	NCDE	32	3
Stock-US	LSTM	64	1
	GRU	64	1
	SeqToSeq (LSTM)	64	1
	SeqToSeq (GRU)	64	1
	ODERNN	64	3
	NCDE	16	2
Stock-China	LSTM	64	1
	GRU	16	1
	SeqToSeq (LSTM)	64	1
	SeqToSeq (GRU)	64	1
	ODERNN	32	1
	NCDE	64	2
USHCN	LSTM	32	1
	GRU	16	1
	SeqToSeq (LSTM)	16	1
	SeqToSeq (GRU)	32	1
	ODERNN	64	2
	NCDE	32	3

that the stock prices for those companies are interconnected, i.e., they are loosely-coupled data. s_{in} and s_{out} for stock datasets are 10 and 4, respectively.

USHCN is a climate dataset that contains monthly average, minimum, and maximum temperature and precipitation in the U.S. states from the U.S. historical climatology network [USHCN, 1987], i.e., $\dim(\mathbf{x}) = 4$. The collection period is 1981–2012. The length of one window is a year, 12 months. We set s_{in} and s_{out} to 10 and 2.

After making the input and output pairs of target models like training procedure in Sec. 3.2, Flu has 20K training pairs, 3K validation pairs, and 3K test pairs. Both Stock-US and Stock-China have 10K training pairs, 1K validation pairs, and 1K test pairs. USHCN has 17K training pairs, 0.5K validation pairs, and 0.5K test pairs.

Hyperparameters in baselines. In this section, we show the best hyperparameters for our baselines for reproducibility in Tables 2 to 8. For our baselines, we have two hyperparameters for the target models, a hidden size and the number of layers. The hidden size is in $\{16, 32, 64\}$ and the number of layers is in $\{1, 2, 3\}$. We set the learning rate to 10^{-3} and the coefficient of L_2 regularization term to 10^{-5}. The size of mini-batch is 256. For *Vanilla*, there is no additional hyperparameter. For *HyperLSTM/GRU*, there are 2 additional hyperparameters, $\dim(\hat{\mathbf{h}})$ and $\dim(\mathbf{a})$, that are the dimensionality of $\hat{\mathbf{h}}$ and $\mathbf{a}$, defined in Ha et al. [2016] or Appendix 5. As $\dim(\mathbf{a}) < \dim(\hat{\mathbf{h}}) < \dim(\mathbf{h}_{\mathcal{B}})$ recommended in Ha et al. [2016], $\dim(\hat{\mathbf{h}})$ is in $\{32, 16, 8\}$, and $\dim(\mathbf{a})$ is in $\{16, 8, 4\}$. For *ANODEV2*, we configure the baseline as in Zhang et al. [2019] and *RevIN* is just adding an adaptation layer, so additional hyperparameters are not needed. *AdaGRU/LSTM* has an additional hyperparameter, λ which is the coefficient of a drift-related loss. In an *ARIMA* case, there are 3 hyperparameters. p, d, and q denote the autoregressive, differences, and moving average components, respectively. p is in {1,2,3}, d in {0,1,2}, q in {1,2,3}.

Hyperparameters in HyperGPA. For *HyperGPA*, the learning rate is set to 10^{-2}, the coefficient of L_2 regularization term to 10^{-6}, and the size of mini-batch to 10^4. The reason why the size of the mini-batch for *HyperGPA* is much larger than that for the baselines is that i) *HyperGPA* generates the parameters of all target models, and ii) all target models are trained simultaneously. Because there are about 30 to 50 target models, the size of mini-batch for one target model is about 256. Each of $\dim(\mathbf{h}_{\mathcal{B}})$ and $n_{\mathcal{B}}$ in *HyperGPA* is set to 16 and 1, respectively.

The hidden size and the number of layers for Γ in Eq. 1 are set to 32 and 2, respectively. The hidden size of NCDEs, $\dim(\mathbf{h}')$ in Eq. (2), is in $\{32, 64, 128\}$. In AGC function, the node embedding size and the output size are set to 32. Φ is a one-layer fully connected layer without bias. The size of initial query vectors in Eq. 3, $\dim(\mathbf{z})$, is in $\{512, 1024, 2048\}$. In Eq. (4), GAT has three hyperparameters: i) number of heads, ii) depth, and iii) hidden size. The number of heads is set to 4, the depth to

Table 4: Hyperparameters for *HyperLSTM/GRU*

Data	Target Model	$\dim(\mathbf{h}_{\mathcal{B}})$	$n_{\mathcal{B}}$	$\dim(\mathbf{a})$	$\dim(\hat{\mathbf{h}})$
Flu	LSTM	64	1	16	8
	GRU	32	1	8	8
	SeqToSeq (LSTM)	64	1	4	8
	SeqToSeq (GRU)	16	1	4	16
Stock-US	LSTM	32	1	4	16
	GRU	16	1	16	8
	SeqToSeq (LSTM)	32	1	8	8
	SeqToSeq (GRU)	64	1	4	8
Stock-China	LSTM	16	1	4	8
	GRU	16	1	4	8
	SeqToSeq (LSTM)	16	1	4	8
	SeqToSeq (GRU)	32	1	4	8
USHCN	LSTM	32	1	4	16
	GRU	64	1	4	8
	SeqToSeq (LSTM)	64	1	4	32
	SeqToSeq (GRU)	64	1	4	8

Table 5: Hyperparameters for *RevIN*

Data	Target Model	$\dim(\mathbf{h}_{\mathcal{B}})$	$n_{\mathcal{B}}$
Flu	LSTM	16	3
	GRU	16	2
	SeqToSeq (LSTM)	16	1
	SeqToSeq (GRU)	32	1
	ODERNN	32	3
	NCDE	64	3
Stock-US	LSTM	16	1
	GRU	16	1
	SeqToSeq (LSTM)	16	1
	SeqToSeq (GRU)	16	1
	ODERNN	32	2
	NCDE	16	3
Stock-China	LSTM	32	1
	GRU	16	2
	SeqToSeq (LSTM)	16	1
	SeqToSeq (GRU)	16	1
	ODERNN	64	3
	NCDE	32	3
USHCN	LSTM	32	1
	GRU	16	3
	SeqToSeq (LSTM)	64	1
	SeqToSeq (GRU)	32	3
	ODERNN	64	2
	NCDE	64	3

Table 6: Hyperparameters for *ANODEV2*

Data	Target Model	$\dim(\mathbf{h}_{\mathcal{B}})$	$n_{\mathcal{B}}$
Flu	ODERNN	64	3
	NCDE	16	3
Stock-US	ODERNN	64	3
	NCDE	64	3
Stock-China	ODERNN	64	2
	NCDE	16	2
USHCN	ODERNN	64	3
	NCDE	32	3

Table 7: Hyperparameters for *AdaLSTM/GRU*

Data	Target Model	$\dim(\mathbf{h}_{\mathcal{B}})$	$n_{\mathcal{B}}$	λ
Flu	LSTM	64	1	1.0
	GRU	64	1	1.0
Stock-US	LSTM	64	1	1.0
	GRU	64	1	1.0
Stock-China	LSTM	64	1	1.0
	GRU	64	1	0.5
USHCN	LSTM	64	1	1.0
	GRU	32	1	1.0

Table 8: Hyperparameters for *ARIMA*

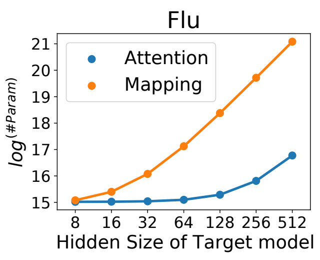
Data	p	d	q
Flu	2	1	1
Stock-US	1	1	2
Stock-China	1	1	1
USHCN	1	1	2

3, and the hidden size to 128. In an attention-based generation method, the number of candidates model C is in $\{3, 5, 10, 20, 48\}$, and the regularization coefficient of MSE_2, λ, is in $\{0, 0.1, 0.01\}$. For input window size K, we use $K = 2$ in `Flu`, `Stock-China`, and `Stock-China`, and $K = 3$ in `USHCN`. Note that all baselines are trained separately for each target model, whereas our method, which internally has a shared multi-task layer, is trained collectively and generates multiple target models' parameters simultaneously.

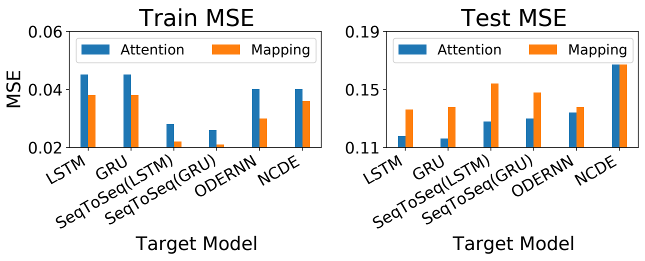
Additional Experiments about Design Choice

Attention-based method or simple mapping method. Fig. 4(a,b) show the advantages of our attention-based parameter generation method, compared to the simple mapping method where $\mathbf{h}_{i,N}$ are directly mapped into the parameters of a target model through linear layers. According to Fig. 4(a), as the hidden size of target models increases, the required number of parameters of *HyperGPA* increases more rapidly in the simple mapping method than in the attention-based method. In addition, our attention-based method is more robust to overfitting as in Fig. 4(b).

Period size K. Fig. 5 shows sensitivity analyses w.r.t. the input period size K. As shown, $K \leq 3$ produces the best outcomes. Feeding too much information leads to sub-optimal outcomes. For this reason, we use 2 or 3 as K.



(a) Required model size



(b) Train and Test MSE

Figure 4: (a) The required model size, and (b) Train and test MSE of the two possible parameter generation methods in `Flu`. 'Attention' means our full model.

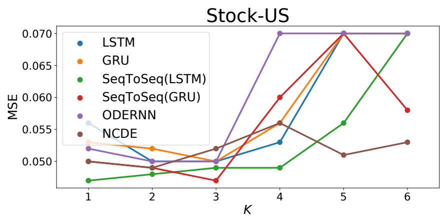


Figure 5: Sensitivity to the input period size K

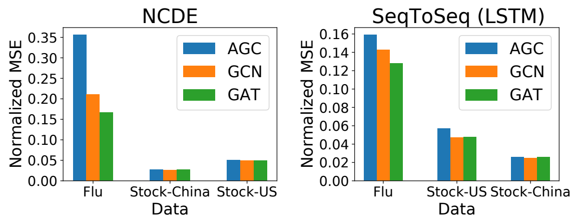


Figure 6: Changing the type of graph functions on the parameter generating layer ($L2$)

Type of graph function in $L2$. Fig. 6 shows the results by varying the graph function of the parameter generating layer ($L2$) in Eq. (4).We test with a graph attention neural network (GAT) [Veličković et al., 2018], a graph convolutional network (GCN) [Kipf and Welling, 2017] and an adaptive graph convolutional network (AGC) [Bai et al., 2020]. In most cases, GAT shows reasonable performance. Therefore, we use GAT as our GNNs in $L2$.

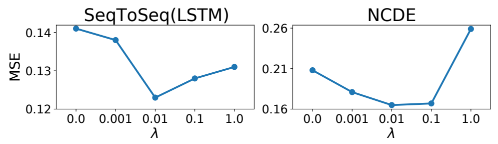


Figure 7: Sensitivity to the coefficient of the second MSE, λ

Regularization coefficient of MSE_2, λ. As we mentioned before, we use the two types of MSE, one of which is measured with $\hat{\boldsymbol{\theta}}_{i,N}$ and the other is with $\tilde{\boldsymbol{\theta}}^{l}_{\hat{c}^{l}_{i,b}}$. When the first and second MSE are denoted as MSE_1 and MSE_2, respectively, our final training loss is $MSE_1 + \lambda MSE_2$. In Fig. 7, we conduct sensitivity studies to λ with *HyperGPA* whose target model is `SeqToSeq(LSTM)` or `NCDE` in `Flu`. The performance of *HyperGPA* improves as λ increases from 0 to some points. However, when λ gets too large, MSE scores deteriorate. Therefore, we use $\{0,0.1,0.01\}$ as the candidate hyperparameters of λ.

Additional Experimental Results

Full experimental results. Full experimental results are shown from Table. 9 to Table. 12. Val.MSE is the MSE value in validation data, which is the criterion for selecting the best models. We also add a simple linear model as baselines because Zeng et al. [2022] shows that the simple linear model outperform complex neural networks, such as transformers.

Forecasting performance in more complex scenarios. To examine the performance of *HyperGPA* more complex scenarios, we increase the dimensionality of input time series. Because it is hard to find time-series datasets with a large number of features which are appropriate for our task, we increase the number of input length for target models, s_{in}. s_{in} is set to twice the originally used s_{in} in each dataset. We examine the performance of *HyperGPA*, *Vanilla*, and *RevIN* for this task where a target model is `GRU`. Table. 13 shows that *HyperGPA* still performs best, compared to baselines.

Table 9: Experimental results for `Flu`

Target Model	Generation Way	Val.MSE	PCC	R^2	Exp.	MSE	MAE
Linear		0.126	0.944	0.870	0.873	0.221	0.231
ARIMA		0.502±0.000	0.695±0.000	0.152±0.000	0.168±0.000	1.091±0.000	0.579±0.000
`LSTM`	*Vanilla*	0.174±0.004	0.910±0.003	0.718±0.023	0.730±0.021	0.367±0.016	0.299±0.008
	HyperLSTM	0.236±0.008	0.852±0.005	0.434±0.038	0.471±0.032	0.582±0.019	0.388±0.010
	RevIN	0.230±0.022	0.917±0.010	0.842±0.017	0.844±0.018	0.506±0.097	0.256±0.007
	AdaLSTM	0.516±0.047	0.814±0.019	0.367±0.099	0.478±0.075	0.740±0.075	0.552±0.036
	HyperGPA	0.068±0.002	0.971±0.001	0.933±0.003	0.933±0.004	0.118±0.004	0.141±0.002
`GRU`	*Vanilla*	0.130±0.003	0.933±0.002	0.807±0.006	0.813±0.006	0.275±0.006	0.250±0.003
	HyperGRU	0.207±0.016	0.863±0.010	0.585±0.035	0.601±0.035	0.520±0.033	0.361±0.011
	RevIN	0.199±0.015	0.938±0.004	0.880±0.007	0.881±0.008	0.379±0.039	0.226±0.007
	AdaGRU	0.405±0.031	0.849±0.009	0.542±0.048	0.613±0.056	0.616±0.035	0.510±0.018
	HyperGPA	0.066±0.002	0.971±0.001	0.938±0.003	0.939±0.003	0.116±0.004	0.140±0.003
`SeqToSeq(LSTM)`	*Vanilla*	0.151±0.003	0.914±0.002	0.733±0.004	0.747±0.004	0.353±0.005	0.290±0.004
	HyperLSTM	0.211±0.015	0.855±0.007	0.502±0.035	0.527±0.032	0.559±0.021	0.372±0.007
	RevIN	0.193±0.007	0.944±0.004	0.888±0.006	0.890±0.006	0.345±0.021	0.219±0.005
	HyperGPA	0.070±0.004	0.969±0.001	0.932±0.006	0.932±0.006	0.128±0.006	0.143±0.004
`SeqToSeq(GRU)`	*Vanilla*	0.114±0.004	0.939±0.001	0.828±0.006	0.834±0.005	0.250±0.005	0.237±0.004
	HyperGRU	0.191±0.008	0.872±0.007	0.574±0.028	0.592±0.028	0.502±0.023	0.347±0.009
	RevIN	0.189±0.010	0.954±0.003	0.905±0.006	0.908±0.006	0.291±0.032	0.204±0.005
	HyperGPA	0.066±0.002	0.968±0.003	0.926±0.012	0.926±0.012	0.130±0.014	0.142±0.004
`ODERNN`	*Vanilla*	0.163±0.011	0.907±0.010	0.761±0.030	0.767±0.027	0.361±0.039	0.301±0.017
	ANODEV2	0.139±0.003	0.926±0.004	0.790±0.007	0.796±0.007	0.298±0.014	0.256±0.009
	RevIN	0.272±0.004	0.905±0.008	0.817±0.011	0.820±0.011	0.549±0.226	0.266±0.014
	HyperGPA	0.075±0.003	0.967±0.004	0.931±0.007	0.932±0.007	0.134±0.016	0.145±0.004
`NCDE`	*Vanilla*	0.137±0.019	0.900±0.011	0.775±0.015	0.781±0.016	0.387±0.042	0.296±0.011
	ANODEV2	0.457±0.002	0.778±0.004	0.041±0.017	0.077±0.018	0.821±0.011	0.480±0.005
	RevIN	0.259±0.012	0.919±0.003	0.843±0.005	0.848±0.005	0.439±0.022	0.251±0.004
	HyperGPA	0.071±0.003	0.959±0.004	0.913±0.009	0.914±0.009	0.167±0.020	0.164±0.008

Visualization. Fig. 8 shows all train, test (ground-truth) and forecast lines by various baselines and *HyperGPA*, and Fig. 9 shows MSE values over time. In almost all cases, *HyperGPA* follows the ground truth line more accurately than others and achieves the lowest MSE across all times. In predicted and MSE lines, lines related to *HyperGPA* are solid, lines related to baseline are dashed, and lines related to ground truth are dotted. We do not include the case of *ARIMA* because *ARIMA* has the worst performance in *Flu*, *Stock-US*, and *Stock-China*.

Table 10: Experimental results for `Stock-US`

Target Model	Generation Way	Val.MSE	PCC	R^2	Exp.	MSE	MAE
Linear		0.041	0.967	0.928	0.928	0.056	0.173
ARIMA		0.205±0.000	0.548±0.000	-0.781±0.000	-0.748±0.000	0.620±0.000	0.609±0.000
`LSTM`	*Vanilla*	0.051±0.001	0.902±0.005	0.569±0.029	0.641±0.022	0.213±0.010	0.307±0.006
	HyperLSTM	0.089±0.005	0.605±0.071	-1.045±0.340	-0.491±0.248	0.751±0.103	0.586±0.038
	RevIN	0.045±0.000	0.966±0.000	0.930±0.000	0.933±0.000	0.063±0.001	0.183±0.001
	AdaLSTM	0.257±0.017	0.787±0.068	0.400±0.231	0.475±0.209	0.379±0.102	0.483±0.062
	HyperGPA	0.034±0.000	0.973±0.000	0.930±0.005	0.933±0.004	0.050±0.002	0.161±0.004
`GRU`	*Vanilla*	0.043±0.001	0.952±0.002	0.830±0.007	0.848±0.006	0.102±0.003	0.226±0.004
	HyperGRU	0.073±0.007	0.772±0.030	-0.167±0.113	0.103±0.086	0.462±0.046	0.453±0.015
	RevIN	0.044±0.000	0.967±0.000	0.933±0.001	0.935±0.000	0.060±0.001	0.178±0.000
	AdaGRU	0.176±0.021	0.875±0.028	0.645±0.073	0.693±0.073	0.233±0.043	0.377±0.045
	HyperGPA	0.034±0.000	0.972±0.001	0.927±0.005	0.930±0.004	0.052±0.002	0.164±0.004
`SeqToSeq(LSTM)`	*Vanilla*	0.042±0.001	0.926±0.003	0.688±0.014	0.744±0.011	0.167±0.005	0.274±0.004
	HyperLSTM	0.077±0.004	0.674±0.052	-0.604±0.181	-0.173±0.120	0.643±0.084	0.528±0.030
	RevIN	0.046±0.001	0.967±0.000	0.932±0.002	0.935±0.001	0.061±0.001	0.180±0.002
	HyperGPA	0.034±0.000	0.973±0.001	0.936±0.002	0.938±0.002	0.048±0.001	0.157±0.002
`SeqToSeq(GRU)`	*Vanilla*	0.039±0.000	0.948±0.003	0.811±0.012	0.835±0.010	0.112±0.006	0.229±0.003
	HyperGRU	0.067±0.002	0.769±0.057	-0.121±0.235	0.141±0.175	0.464±0.092	0.444±0.036
	RevIN	0.044±0.001	0.967±0.001	0.933±0.002	0.935±0.002	0.060±0.002	0.177±0.002
	HyperGPA	0.034±0.000	0.973±0.001	0.935±0.007	0.938±0.005	0.049±0.003	0.159±0.006
`ODERNN`	*Vanilla*	0.053±0.002	0.894±0.014	0.659±0.028	0.697±0.023	0.200±0.015	0.300±0.008
	ANODEV2	0.043±0.001	0.940±0.009	0.801±0.024	0.822±0.023	0.120±0.013	0.236±0.008
	RevIN	0.048±0.001	0.964±0.000	0.926±0.001	0.929±0.001	0.068±0.001	0.188±0.001
	HyperGPA	0.034±0.000	0.972±0.000	0.931±0.001	0.933±0.001	0.050±0.001	0.162±0.002
`NCDE`	*Vanilla*	0.045±0.001	0.929±0.007	0.805±0.029	0.822±0.024	0.130±0.015	0.247±0.010
	ANODEV2	0.152±0.000	0.850±0.001	0.551±0.003	0.555±0.003	0.244±0.002	0.376±0.002
	RevIN	0.044±0.001	0.967±0.000	0.934±0.001	0.935±0.000	0.060±0.001	0.177±0.001
	HyperGPA	0.034±0.000	0.973±0.001	0.934±0.004	0.936±0.003	0.049±0.002	0.159±0.003

Table 11: Experimental results for `Stock-China`

Target Model	Generation Way	Val.MSE	PCC	R^2	Exp.	MSE	MAE
Linear		0.068	0.984	0.966	0.967	0.032	0.130
ARIMA		0.478±0.000	0.840±0.000	0.570±0.000	0.682±0.000	0.439±0.000	0.503±0.000
`LSTM`	*Vanilla*	0.084±0.002	0.975±0.000	0.945±0.001	0.946±0.001	0.050±0.001	0.160±0.002
	HyperLSTM	0.188±0.019	0.949±0.003	0.879±0.009	0.881±0.008	0.103±0.007	0.226±0.004
	RevIN	0.081±0.001	0.977±0.001	0.954±0.002	0.955±0.002	0.049±0.002	0.152±0.002
	AdaLSTM	0.359±0.068	0.834±0.037	0.565±0.183	0.571±0.180	0.321±0.064	0.460±0.039
	HyperGPA	0.059±0.001	0.987±0.000	0.972±0.001	0.972±0.001	0.026±0.001	0.113±0.002
`GRU`	*Vanilla*	0.089±0.003	0.982±0.001	0.959±0.002	0.960±0.002	0.037±0.002	0.138±0.003
	HyperGRU	0.156±0.021	0.963±0.002	0.915±0.003	0.917±0.003	0.075±0.002	0.194±0.004
	RevIN	0.081±0.000	0.981±0.000	0.962±0.000	0.962±0.000	0.040±0.000	0.143±0.001
	AdaGRU	0.199±0.015	0.913±0.009	0.783±0.032	0.786±0.032	0.170±0.017	0.327±0.023
	HyperGPA	0.058±0.002	0.987±0.000	0.972±0.001	0.972±0.001	0.026±0.001	0.114±0.003
`SeqToSeq(LSTM)`	*Vanilla*	0.084±0.002	0.978±0.001	0.951±0.002	0.952±0.002	0.045±0.001	0.151±0.002
	HyperLSTM	0.163±0.017	0.952±0.023	0.889±0.054	0.892±0.051	0.097±0.045	0.205±0.022
	RevIN	0.079±0.001	0.979±0.001	0.958±0.002	0.959±0.002	0.044±0.002	0.144±0.002
	HyperGPA	0.058±0.001	0.987±0.000	0.973±0.001	0.973±0.001	0.026±0.001	0.112±0.001
`SeqToSeq(GRU)`	*Vanilla*	0.072±0.002	0.983±0.000	0.963±0.001	0.964±0.001	0.035±0.001	0.131±0.002
	HyperGRU	0.150±0.018	0.964±0.004	0.920±0.009	0.922±0.008	0.073±0.008	0.187±0.007
	RevIN	0.078±0.001	0.981±0.001	0.962±0.002	0.963±0.002	0.039±0.003	0.138±0.003
	HyperGPA	0.058±0.001	0.988±0.000	0.973±0.001	0.973±0.001	0.025±0.000	0.112±0.001
`ODERNN`	*Vanilla*	0.092±0.013	0.972±0.004	0.938±0.009	0.939±0.009	0.056±0.007	0.171±0.013
	ANODEV2	0.079±0.002	0.982±0.001	0.961±0.001	0.962±0.001	0.037±0.001	0.139±0.002
	RevIN	0.086±0.001	0.978±0.001	0.955±0.002	0.956±0.002	0.048±0.002	0.154±0.002
	HyperGPA	0.058±0.001	0.987±0.000	0.972±0.001	0.972±0.001	0.026±0.001	0.115±0.003
`NCDE`	*Vanilla*	0.081±0.004	0.980±0.001	0.959±0.002	0.960±0.002	0.040±0.001	0.142±0.001
	ANODEV2	0.335±0.001	0.929±0.001	0.832±0.003	0.838±0.003	0.141±0.001	0.292±0.002
	RevIN	0.088±0.001	0.981±0.000	0.962±0.000	0.962±0.000	0.040±0.001	0.142±0.001
	HyperGPA	0.060±0.003	0.987±0.001	0.971±0.001	0.971±0.001	0.027±0.001	0.116±0.003

Table 12: Experimental results for `USHCN`

Target Model	Generation Way	Val.MSE	PCC	R^2	Exp.	MSE	MAE
Linear		0.322	0.824	0.071	0.174	0.244	0.347
ARIMA		0.277±0.000	0.838±0.000	0.284±0.001	0.394±0.001	0.232±0.000	0.313±0.000
`LSTM`	*Vanilla*	0.298±0.001	0.841±0.001	0.152±0.026	0.270±0.021	0.239±0.003	0.345±0.004
	HyperLSTM	0.320±0.003	0.824±0.003	0.075±0.009	0.191±0.009	0.249±0.004	0.350±0.003
	RevIN	0.779±0.029	0.693±0.009	-0.019±0.054	0.225±0.035	0.589±0.015	0.610±0.010
	AdaLSTM	0.679±0.013	0.594±0.015	-0.533±0.259	-0.327±0.222	0.595±0.026	0.613±0.012
	HyperGPA	0.299±0.003	0.844±0.002	0.127±0.040	0.245±0.040	0.221±0.003	0.313±0.009
`GRU`	*Vanilla*	0.298±0.001	0.840±0.001	0.156±0.014	0.264±0.012	0.232±0.001	0.334±0.001
	HyperGRU	0.313±0.002	0.832±0.002	0.124±0.028	0.239±0.022	0.244±0.004	0.346±0.005
	RevIN	0.599±0.016	0.699±0.008	0.057±0.070	0.178±0.050	0.465±0.007	0.522±0.006
	AdaGRU	0.545±0.017	0.676±0.018	-0.171±0.310	-0.035±0.269	0.472±0.026	0.532±0.014
	HyperGPA	0.299±0.003	0.841±0.003	0.142±0.025	0.266±0.009	0.229±0.004	0.323±0.003
`SeqToSeq(LSTM)`	*Vanilla*	0.299±0.001	0.835±0.002	0.157±0.015	0.261±0.011	0.236±0.003	0.335±0.004
	HyperLSTM	0.316±0.002	0.826±0.003	0.141±0.013	0.244±0.014	0.243±0.004	0.343±0.004
	RevIN	0.730±0.023	0.682±0.016	-0.059±0.075	0.143±0.052	0.585±0.014	0.600±0.009
	HyperGPA	0.296±0.005	0.843±0.002	0.242±0.043	0.344±0.043	0.220±0.007	0.307±0.006
`SeqToSeq(GRU)`	*Vanilla*	0.294±0.001	0.839±0.001	0.149±0.016	0.256±0.013	0.232±0.001	0.335±0.002
	HyperGRU	0.308±0.003	0.831±0.001	0.169±0.013	0.265±0.014	0.236±0.003	0.336±0.003
	RevIN	0.650±0.007	0.678±0.003	-0.087±0.056	0.090±0.047	0.519±0.006	0.563±0.001
	HyperGPA	0.302±0.002	0.844±0.005	0.182±0.083	0.287±0.083	0.222±0.004	0.315±0.006
`ODERNN`	*Vanilla*	0.300±0.001	0.840±0.001	0.192±0.023	0.297±0.018	0.235±0.004	0.339±0.008
	ANODEV2	0.296±0.001	0.842±0.001	0.181±0.029	0.291±0.023	0.233±0.004	0.337±0.006
	RevIN	1.162±0.066	0.529±0.021	-0.784±0.248	-0.327±0.180	0.855±0.046	0.757±0.019
	HyperGPA	0.291±0.001	0.840±0.004	0.182±0.077	0.286±0.078	0.226±0.007	0.319±0.011
`NCDE`	*Vanilla*	0.304±0.002	0.829±0.002	0.188±0.038	0.294±0.031	0.234±0.004	0.322±0.002
	ANODEV2	0.754±0.001	0.616±0.001	-2.257±0.065	-1.924±0.056	0.483±0.002	0.554±0.001
	RevIN	1.052±0.014	0.467±0.015	-0.920±0.086	-0.724±0.087	0.713±0.008	0.687±0.006
	HyperGPA	0.305±0.008	0.837±0.003	0.134±0.062	0.246±0.073	0.227±0.004	0.316±0.006

Table 13: Experimental results in complex scenarios with large s_{in}

Dataset	Generation Way	Val.MSE	PCC	R^2	Exp.	MSE	MAE
`Flu`	*Vanilla*	0.126	0.933	0.803	0.811	0.277	0.253
	RevIN	0.389	0.863	0.733	0.746	0.759	0.358
	HyperGPA	0.066	0.971	0.934	0.934	0.118	0.141
`Stock-US`	*Vanilla*	0.045	0.952	0.822	0.841	0.104	0.231
	RevIN	0.066	0.952	0.900	0.905	0.092	0.227
	HyperGPA	0.035	0.972	0.921	0.926	0.054	0.170
`Stock-China`	*Vanilla*	0.091	0.981	0.956	0.957	0.040	0.142
	RevIN	0.140	0.968	0.934	0.937	0.070	0.182
	HyperGPA	0.060	0.987	0.971	0.972	0.026	0.115

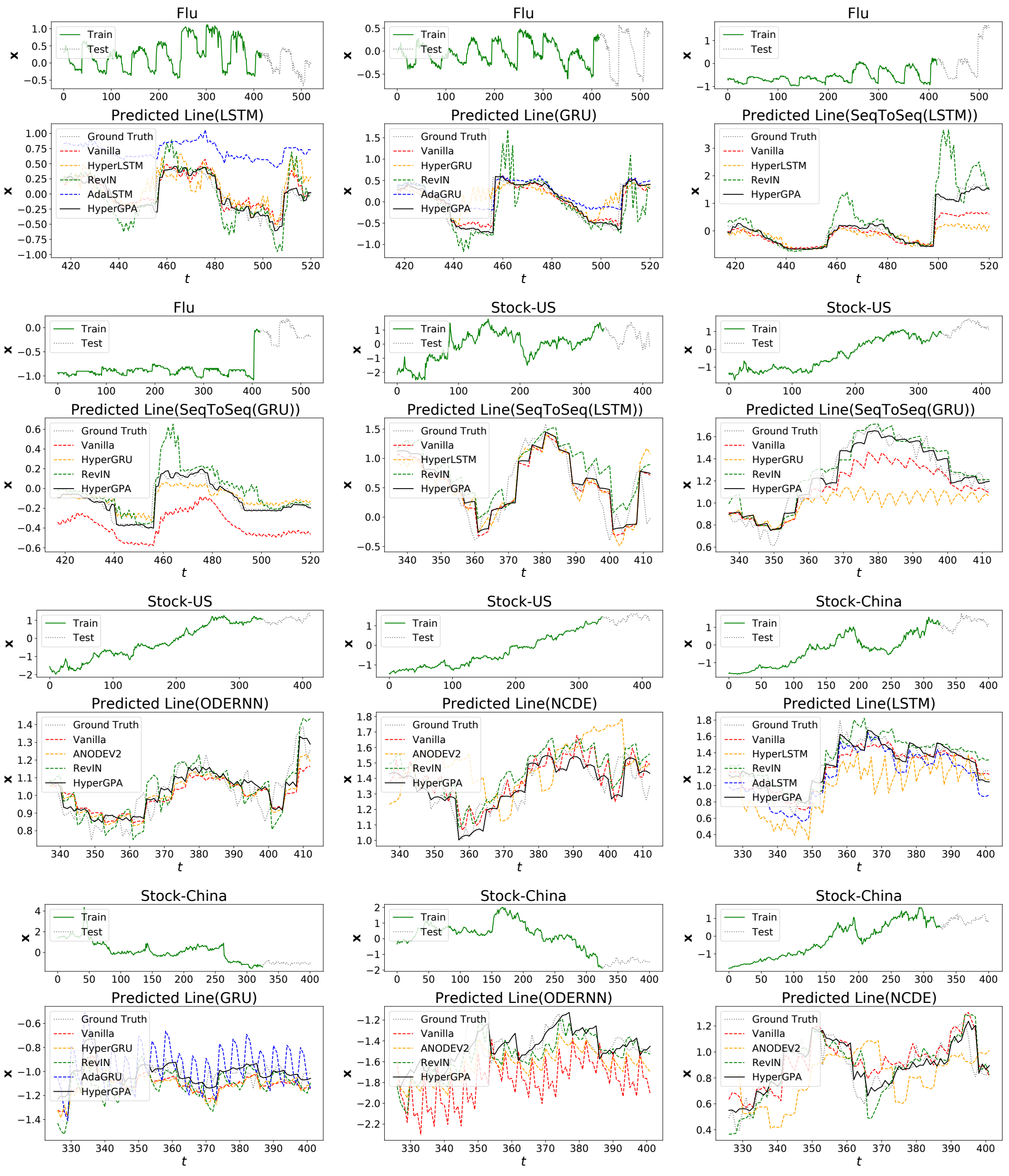


Figure 8: Forecasting visualizations of the baselines and *HyperGPA*

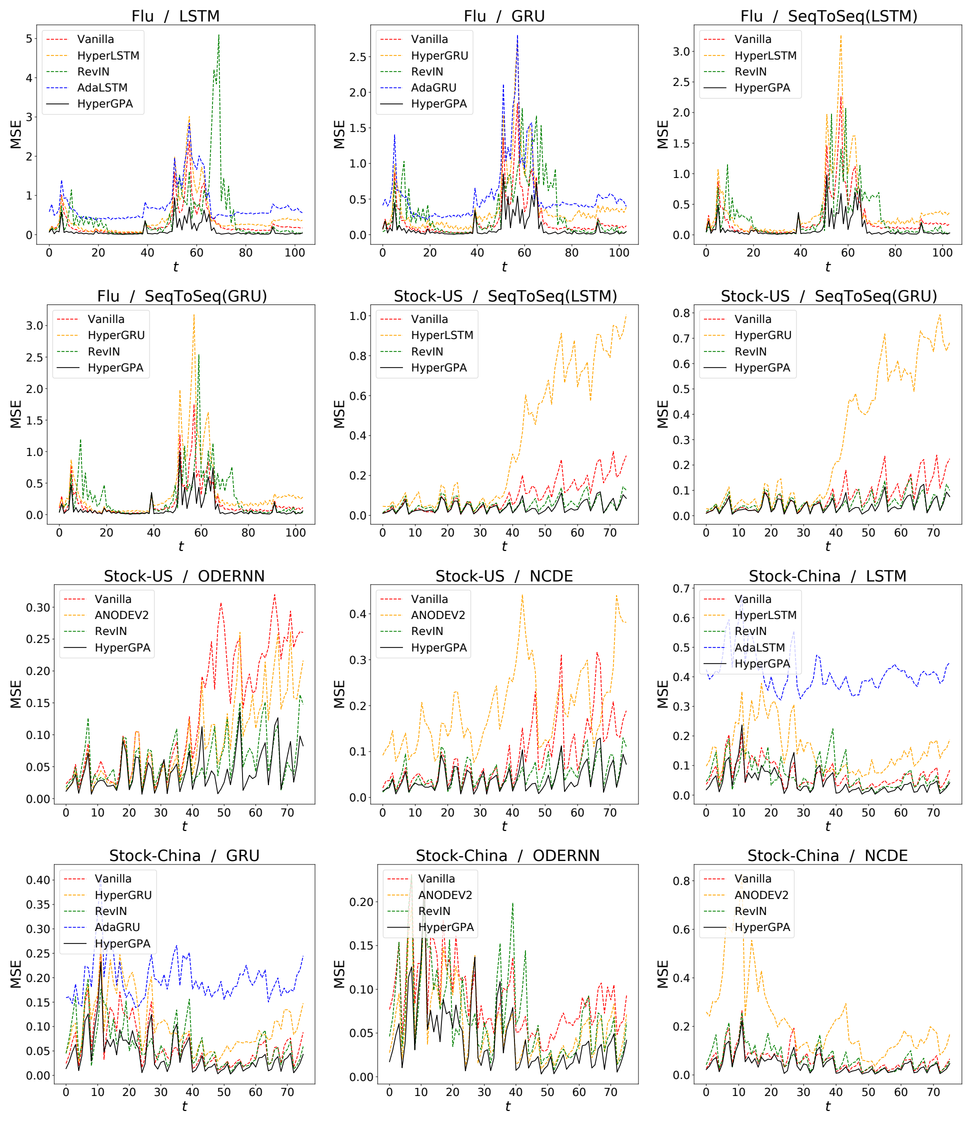


Figure 9: MSE over time