Combating Distribution Shift for Accurate Time Series Forecasting via Hypernetworks

1st Wenying Duan  
Nanchang University  
Nanchang, China  
wenyingduan@ncu.edu.cn

2nd Xiaoxi He  
University of Macau  
Macau  
cloud.de.he@gmail.com

3rd Lu Zhou  
The Hong Kong Polytechnic University  
Hong Kong  
lu.lz.zhou@polyu.edu.hk

4th Lothar Thiele  
ETH Zurich  
Zurich, Switzerland  
thiele@ethz.ch

5th Hong Rao*  
Nanchang University  
Nanchang, China  
raohong@ncu.edu.cn

Abstract—Time series forecasting has widespread applications in urban life ranging from air quality monitoring to traffic analysis. However, accurate time series forecasting is challenging because real-world time series suffer from the distribution shift problem, where their statistical properties change over time. Despite extensive solutions to distribution shifts in domain adaptation or generalization, they fail to function effectively in unknown, constantly-changing distribution shifts, which are common in time series. In this paper, we propose Hyper Time-Series Forecasting (HTSF), a hypernetwork-based framework for accurate time series forecasting under distribution shift. HTSF jointly learns the time-varying distributions and the corresponding forecasting models in an end-to-end fashion. Specifically, HTSF exploits the hyper layers to learn the best characterization of the distribution shifts, generating the model parameters for the main layers to make accurate predictions. We implement HTSF as an extensible framework that can incorporate diverse time series forecasting models such as RNNs. Extensive experiments on 7 benchmarks demonstrate that HTSF achieves state-of-the-art performances.

Index Terms—hypernetworks, time series forecasting, distribution shift

I. INTRODUCTION

Time-series forecasting is crucial for various data analytics domains, including air quality monitoring [1], renewable energy production [2], human activity recognition [3], urban traffic analysis [4] etc. The past years have witnessed diverse time-series forecasting models ranging from the conventional statistical approaches e.g., auto-regressive integrated moving average (ARIMA) [5] to the more recent deep learning based models such as recurrent neural networks (RNNs) [6], transformers, [7] and their variants [8]–[11]. As deep learning based models make few assumptions on the temporal structures of time series, they are preferable in modeling complex, long-sequence time series. They have demonstrated a notable improvement in prediction accuracy than statistical models [8].

A unique challenge in time series forecasting is that they are often non-stationary, resulting in temporal shifts in their distributions. This distribution shift phenomenon causes discrepancies between the distributions of the training and the testing data, which degrades the performance of forecasting models [12], [13]. Such distribution shift also exists among the input sequences of the training data, which makes it challenging to train a model that generalizes well to unknown data [14], [15]. Figure 1 illustrates the distribution shift problem in time series forecasting. Consider three sliding windows 1, 2, and 3 to segment the time series for training. The probability distributions of the time series $p_1, p_2, p_3$ may vary across windows, and they are likely to differ from the distribution $p_f$ in the unseen forecast window. That is, $p_1 \neq p_2 \neq p_3 \neq p_f$, which makes accurate forecasting for window $p_f$ challenging.

Distribution shift is usually tackled by domain generalization [16] or domain adaptation [15], [17]. The idea is to learn the common knowledge transferable between domains despite the differences in their distributions [16]. However, it is non-trivial to apply these techniques to time series [13], [18]. This is because the distributions in time series change constantly (e.g., $p_1, p_2, p_3$ in Figure 1), and it is unknown how to best characterize the distributions upon which to learn the common

*Corresponding Author
knowledge [13]. The state-of-the-art, ADARNN [13] proposes to characterize the worst-case distribution shift in time series leveraging the principle of maximum entropy. Nevertheless, this approach is sub-optimal because (i) the training data may not always contain the worst-case distribution shift; and (ii) it under-utilizes the diversity of the distribution shifts in the time series.

In this paper, we propose a hypernetwork-based framework called Hyper Time-Series Forecasting (HTSF) to mitigate the distribution shift problem for accurate time series forecasting. The idea is to jointly learn the common knowledge and the corresponding distributions with an end-to-end hypernetwork architecture. As shown in Figure 1, a hypernetwork consists of hyper and main layers, where the hyper layers generate model parameters $W_f$ conditioned on the inputs $x_f$ for the main layers to make predictions $y_f$. We exploit the hyper layers to learn the best characterization of the distribution shifts and the main layers to learn the corresponding forecast model. Our strategy is advantageous in the following aspects. (i) The end-to-end learning scheme eliminates the assumptions on the worst-case distribution shift in the training data and makes full use of the various distribution shifts in the training data. (ii) The hypernetwork architecture is capable of modeling complex feature representations in both the distributions and the common transferable knowledge, and it applies to various deep learning based time series models.

Our main contributions are summarized as follows:

- We propose HTSF, a hypernetwork-based framework for accurate time series forecasting. To the best of our knowledge, it is the first study that addresses the distribution shift problem in time series via hypernetworks.
- HTSF offers a generic, extensible, and end-to-end solution to the distribution shift problem in time series that makes no assumptions on the training data and applies to diverse deep learning based time series models such as variants of RNNs.
- We evaluate HTSF on 7 time series forecasting benchmarks. Experimental results show that HTSF achieves state-of-the-art accuracy in time series forecasting. It also yields comparable performance on spatiotemporal data to the state-of-the-art graph neural network based solutions [19], [20].

II. RELATED WORK

Our work is related to the following categories of research.

A. Time Series Forecasting

For time series forecasting, we distinguish between statistic models and deep learning based models. Statistic models [21] are interpretable and theoretically sound. But they often suffer from heavy crafting on data pre-processing and labor-intensive feature engineering, which may fail to capture complex patterns in time series. Recently, deep learning based models have attracted increasing attention due to their outstanding performance in time series analysis. For example, RNN-based models [2], [8], [22] prove effective in capturing both short-term and long-term patterns. Transformer-based models [11], [23] have shown remarkable performance for extremely long sequences. However, they are not robust to the distribution shift problem and suffer from performance degradation [13], [18].

B. Distribution Shift

Domain adaptation [15], [17] and domain generalization [16] are generic solutions to the distribution shift problem. Domain adaptation focuses on the disparity difference between the source and target domain, whereas domain generalization aims to learn a generalizable model from various source domains. However, it is non-trivial to apply these techniques to time series. For example, some research reports [24] that domain adversarial learning, a prevailing domain adaptation technique, may be inappropriate for regression, while time series forecasting is often a regression task. Furthermore, since the distributions in time series change constantly, it is not straightforward how to define multiple source domains in non-stationary time series [13].

A few pioneer studies [13], [18] explore solving the distribution shift problem in time series forecasting. RevIN [18] proposes an instance-level normalization technique to reduce the discrepancy in data distributions such as mean and variance. Our work is orthogonal to RevIN by adopting representation learning to enhance the generalization of forecasting models. Our work is most related to ADARNN [13], which first attempts to characterize the worst-case distribution shift in time series and then matches the distributions to learn a generalizable model. We advance ADARNN by eliminating the worst-case assumption and fully exploiting the diversity of the distribution shifts in the training data.

C. Hypernetworks

A hypernetwork [25] utilizes a primary neural network $g$ to generate weights $\theta_f$ for a second network $f$. The network $f$ with the weights $\theta_f$ can then be applied for specific inference tasks. Hypernetworks have demonstrated superior performances on many benchmarks [26], [27], due to their ability to adapt $f$ for different inputs, which allows the hypernetwork to model tasks more effectively [28]. Hypernetworks have been applied in few-shot learning [29], continual learning [27], efficient parameter fine-tuning [30], [31] etc. A few studies have also introduced hypernetworks in time series forecasting such as traffic prediction [32]. However, it does not explicitly account for the distribution shift problem, and thus results in sub-optimal accuracy. To the best of our knowledge, we are the first to adopt hypernetworks to tackle the distribution shift problem in time series forecasting.

III. PROBLEM STATEMENT

We consider a standard multi-variate time series forecasting problem as follows. A time series $X$ with the corresponding label sequence $Y$, is segmented into a set of sequences with
labels $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$, where $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x \times T_x}$ is a subsequence of $\mathcal{X}$, $\mathbf{y}^{(i)} \in \mathbb{R}^{d_y \times T_y}$ is the corresponding label form $\mathcal{Y}$. $N$, $d_x$, $d_y$, $T_x$, and $T_y$ denote the number of sequences, the dimension of input variables, the dimension of label variables, the input length and the prediction length, respectively. Our objective is to learn a model $\mathcal{M}: x_i \rightarrow y_i$ from $\mathcal{D}_{\text{train}}$ for unseen sequences in $\mathcal{D}_{\text{test}}$, where $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ are the training and test sets from $\mathcal{D}$.

We aim to learn model $\mathcal{M}$ in presence of distribution shift in the time series, which is described by the following assumptions.

- There is discrepancy between the marginal probability distribution $P_{\mathcal{D}_{\text{train}}}$ of the training set and the distribution $P_{\mathcal{D}_{\text{test}}}$ of the test set, i.e., $P_{\mathcal{D}_{\text{train}}} \neq P_{\mathcal{D}_{\text{test}}}$. Note that $P_{\mathcal{D}_{\text{train}}} (Y|X) = P_{\mathcal{D}_{\text{test}}} (Y|X)$ because the underlying laws governing the inputs and the outputs of the time series usually stay the same [13].
- The marginal probability distribution of each $\mathbf{x}^{(i)}$ from $\mathcal{D}_{\text{train}}$ may differ, although their conditional distributions be the historical data of $x_i$. Let $\mathbb{X}$ denote the number of sequences, the hyper layers take all $\mathbf{x}^{(i)}$ from $\mathcal{S}$ as inputs and output a set of representations of the historical marginal probability distributions. These representations and input at the current time step from $\mathbf{x}^{(i)}$ are then combined through the attention mechanism to generate the weights for the main layers at the next time step. Finally, the main layers make predictions as conventional time series forecasting methods.

Note that our method makes full use of the various distribution shifts in historical data $\mathcal{S}$ and aggregates the common transferable knowledge from $\mathcal{S}$ to mitigate the distribution shift problem for accurate time series forecasting. In comparison, prior proposals either ignore the distribution shift problem [8], [9], [11] or under-utilize the diverse distribution shifts in the time series [13].

### IV. Method

This section presents HTSF, our hypernetwork-based framework for time series forecasting under distribution shift. HTSF is a model-agnostic framework that can incorporate various deep learning based time series forecasting models such as RNNs. For ease of presentation, we explain HTSF using a gated recurrent unit (GRU) as an example forecasting model.

#### A. HTSF Overview

Figure 2 depicts the overview of HTSF with GRUs as the underlying forecasting model, which we call HyperGRU. It mainly consists of two modules.

- **Hyper Layers**: These layers learn to characterize the distributions from the historical sequences. The learned distribution characterizations are then used to generate the weights of the main layers. We employ bidirectional GRU (BiGRU) for the hyper layers (see Sec. IV-B).

- **Main Layers**: These layers adapt their weights according to the learned distribution characterizations from the hyper layers as well as their own internal states through an attention mechanism. As a result, the main layers can leverage the common knowledge extracted from the diverse distribution shifts to make accurate predictions. The core part of the main layers is a simple GRU cell (see Sec. IV-C).

The hyper and main layers of HyperGRU interact as follows. Let $\mathcal{X}^{(i)}$ be the historical data of $x^{(i)}$, $\mathcal{S} = \{\mathbf{x}^{(i)}\}_{i=1}^N \subset \mathcal{X}^{(i)}$ is the historical dataset generated by $\mathcal{X}^{(i)}$ with a sliding window of a fixed length $T$, where $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x \times T_x}$, $L = |\mathcal{X}^{(i)}| - T + 1$. HyperGRU takes pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ as inputs. Specifically, the hyper layers take all $\hat{\mathbf{x}}^{(i)}$ from $\mathcal{S}$ as inputs and output $\mathbf{y}^{(i)}$ as output. These representations and input at the current time step from $\mathbf{x}^{(i)}$ are then combined through the attention mechanism to generate the weights for the main layers at the next time step. Finally, the main layers make predictions as conventional time series forecasting methods.

#### B. Hyper Layers

The hyper layers are responsible to learn the complex feature representations of various distribution shifts. We use BiGRU to collectively encode all the historical data $\mathcal{S} = \{\mathbf{x}^{(i)}\}_{n=1}^N$.

- We first apply an 1D average pooling with kernel size $k$ and stride $k$ over $\mathbf{x}^{(i)}$ on time dimension to down sample $\mathbf{x}^{(i)}$ into its $\frac{T}{k}$ slice:

$$\mathbf{x}^{(i)} = \text{AvgPool1D} (\mathbf{x}^{(i)})$$  \hspace{1cm} (1)

where $\mathbf{x}^{(i)} \in \mathbb{R}^{d_x \times \frac{T}{k}}$. (1) reduces the memory usage while preserving most of the information in the historical data [11]. We empirically show that such downsampling has little impact on the prediction accuracy (see Sec. IV-B).

- Then we feed $\mathbf{x}^{(i)}$ into a BiGRU to generate the state sequence $\mathbf{h}^{(i)} \in \mathbb{R}^{d_h \times \frac{T}{k}}$:

$$\mathbf{h}^{(i)} = \left\{ h^{(i)}_{n,(i)} \right\}_{t=1}^{\frac{T}{k}} = \text{BiGRU} (\mathbf{x}^{(i)})$$  \hspace{1cm} (2)
where $h_{n,t}^{(i)} \in \mathbb{R}^{d_h}$ is a hidden state with feature dimension $d_h$. Note that $h_{n,t}^{(i)}$ can be considered as the distribution characterization for $x_{n,t}^{(i)}$.

The hyper layer explicitly models and characterizes the distributions from the historical data. Such information is then utilized to generate the weights of the main layers, as explained below.

### C. Main Layers

The main layers can adaptively change their parameters conditioned on $h_{n,t}^{(i)}$ and $x^i$. The core part of a main layer is a simple GRU cell. To illustrate its weights generation process, we first present the standard formulation of the GRU cell:

$$
\begin{align*}
    r_t &= \sigma(W_{rx} s_{t-1}^{(i)} + W_{hr} h_{t-1}^{(i)} + b_r) \\
    z_t &= \sigma(W_{xz} s_{t-1}^{(i)} + W_{hz} h_{t-1}^{(i)} + b_z) \\
    n_t &= \tanh(W_{xn} s_{t-1}^{(i)} + r_t \otimes (W_{hn} h_{t-1}^{(i)} + b_n)) \\
    s_t^{(i)} &= (1 - z_t) \odot n_t + z_t \odot s_{t-1}^{(i)}
\end{align*}
$$

where $s_t^{(i)} \in \mathbb{R}^{d_h}$ denotes input of the GRU cell from $x_t^{(i)}$ at time step $t$, $s_t^{(i)} \in \mathbb{R}^{d_h}$ is the hidden state at $t$, $\sigma$ is sigmoid function, $W_{rx}$, $W_{xz}$, $W_{xn}$, $W_{hr}$, $W_{hz}$, $W_{hn}$, $b_r$, $b_z$ and $b_n$ are learnable parameters.

The initial hidden state $s_0^{(i)}$ is computed as follows:

$$
    s_0^{(i)} = W_{inist} h_{n,1}^{(i)} + b_{inist}
$$

where $h_{n,1}^{(i)}$ is the last state in $h_n^{(i)}$, $W_{inist}$ and $b_{inist}$ are learnable parameters.

The weights of the GRU cell at current time step $t$ are controlled by a vector $v_t$ that varies dynamically with $t$:

$$
W_{Ix} = W_{hIx} \nu_t, W_{Iz} = W_{hIz} \nu_t, W_{Ih} = W_{hIh} \nu_t
$$

where $W_{hIx}$, $W_{hIz}$ and $W_{hIh}$ are learnable parameters. $W_{Ix} \in \mathbb{R}^{d_h \times d_x}$, $W_{Iz} \in \mathbb{R}^{d_h \times d_h}$, and $W_{Ih} \in \mathbb{R}^{d_h \times d_h}$ are the weights of the GRU cell, $v^t$ is determined by the previous state $s_{t-1}^{(i)}$, and the outputs of the hyper layers $h_{n}^{(i)}$, which is computed by the attention mechanism:

$$
\begin{align*}
    v_t &= W_v c_t \\
    c_t &= \frac{1}{T} \sum_{p=1}^{T} \alpha_p h_{n,p}^{(i)} \\
    \alpha_p &= \frac{\exp(\text{score}(s_{t-1}^{(i)}, h_{n,p}^{(i)}))}{\sum_{p=1}^{T} \exp(\text{score}(s_{t-1}^{(i)}, h_{n,p}^{(i)}))}
\end{align*}
$$

where score(•) is a score function, according to Luong attention [33]. The score function is given by:

$$
\text{score}(s_{t-1}^{(i)}, h_{n,p}^{(i)}) = V^T \tanh(W_s s_{t-1}^{(i)} + W_h h_{n,p}^{(i)} + b_h)
$$

where $V$, $W_s$, $W_h$, and $b_h$ are learnable parameters.

As mentioned above, all the parameters of the main layer are generated by $W_{Ix}$, $W_{Iz}$, and $W_{Ih}$:

$$
\begin{align*}
    (W_{hr}, W_{hIx}, W_{hIz}) &= \text{Chunk}(W_{hIx}) \\
    (W_{xz}, W_{Iz}, W_{Ih}) &= \text{Chunk}(W_{Ih}) \\
    (b_r, b_z, b_n) &= \text{Chunk}(W_{Ih})
\end{align*}
$$

where $W_{hr}, W_{hIz}, W_{hIh} \in \mathbb{R}^{d_h \times d_x}$, $d_x \times 3d_h = d_{lh}$ and $W_{Ix}$, $W_{xz}$, $W_{Iz}$, $W_{Ih} \in \mathbb{R}^{d_h \times d_h}$, $d_h \times 3d_h = d_{lx}$ and $b_r, b_z, b_n \in \mathbb{R}^{d_h}$, $3d_h = d_{lh}$. Chunk(•) is the operation to split a tensor into a specific number of equal-sized chunks. Through (6), we can adaptively attach different information in $h_{n,t}^{(i)}$ to the state of the GRU cell at the current time step $t$ to dynamically adjust the weights of the main layers. By integrating (3), (5) and (6), we can dynamically generate weights of the main layers.

### D. Training

According to Sec. IV-B and Sec. IV-C, we implement HyperGRU with a simple prediction layer (e.g., sigmoid function) that takes $\hat{x}_n^{(i)}$ and $x_t^{(i)}$ as inputs and outputs the prediction:

$$
\hat{y}_n^{(i)} = \text{HyperGRU}(x_t^{(i)}, \hat{x}_n^{(i)})
$$

The loss of the $i$-th pair $(x_t^{(i)}, y_t^{(i)})$ is calculated as follows:

$$
\mathcal{L}_{\text{pred}}(\theta) = \frac{1}{L} \sum_{n=1}^{L} \text{criterion}(\hat{y}_n^{(i)}, y_t^{(i)})
$$

where criterion(•, •) is an objective function, and $\theta$ denotes the learnable parameters of HyperGRU. Then we perform training using (10).

### E. How HTSF Handles Distribution Shift

Given any two different pairs $(\hat{x}_{n,m}^{(i)}, x_{n+m}^{(i)})$ and $(\hat{x}_{n,L}^{(i)}, x_{n+L}^{(i)})$, where $x_{n+m}^{(i)}, \hat{x}_{n+m}^{(i)} \in S$, $1 \leq n \neq m \leq L$. The hyper layer takes $\hat{x}_{n}^{(i)}$ and $\hat{x}_{n}^{(i)}$ as inputs and outputs the learned distribution characterization $h_{n}^{(i)}$ and $h_{n}^{(i)}$. As the parameters of the main layer is conditioned on $h_{n}^{(i)}$ and $x^i$, HTSF forces $(\hat{x}_{n}^{(i)}$ and $x_{n}^{(i)})$ to predict the same value $y_t^{(i)}$ by (9) and (10), i.e., minimizing the following objectives:

$$
\begin{align*}
    \mathcal{L}_1 &= \text{criterion}(\hat{y}_{n}^{(i)}, y_t^{(i)}) \\
    \mathcal{L}_2 &= \text{criterion}(\hat{y}_{m}^{(i)}, y_t^{(i)})
\end{align*}
$$

where $\hat{y}_{n}^{(i)}$ and $\hat{y}_{m}^{(i)}$ are computed by (9).

When minimizing (11), HTSF is also trying to reduce the distribution diversities among $\hat{x}_{n}^{(i)}$ and $\hat{x}_{n}^{(i)}$, thus learning the common knowledge shared by different distribution shifts. Recent work [13] has shown that common knowledge can improve the model’s generalization ability.

### F. Extensions to Other Models

As mentioned, HTSF is a model-agnostic framework that can incorporate various time series forecasting models. We briefly explain how to extend HTSF to other RNNs such as vanilla RNN, LSTM [34], ConvLSTM [35] et al.

Extending HTSF to other RNNs is very straightforward. Take the LSTM as an example, we can use (6) to calculate $v_t$, and use $v_t$ to generate the weights of memory cell, input gate, forget gate and output gate. The weights of other RNNs can be generated in a similar way.
V. EXPERIMENTS

In this section, we evaluate the performance of HTSF on 7 benchmarks and different time series models covering short sequence time series forecasting (Sec. V-A) and graph-based data spatiotemporal forecasting (Sec. V-B). Our code is available at https://github.com/wenyingduan/HTSF.

A. General Time Series Forecasting

We test HTSF for time series forecasting on three datasets from diverse application domains.

1) Datasets and Metrics:

- Air Quality [1]: This dataset contains hourly air quality data collected from 12 stations in Beijing, China, from March 2013 to February 2017. We use data from the same four stations (Dongsi, Tiantan, Nongzhangguan, and Dingling) and the same six features (PM2.5, PM10, SO2, NO2, CO, and O3) as [13]. We use the data from 01/03/2013 to 30/06/2016 for training, data from 01/07/2016 to 31/10/2016 for validation, and data from 02/11/2016 to 28/02/2017 for testing. The Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are used as the performance metrics.

- Electricity Consumption [36]: This dataset contains household electricity power consumption measurements of 2,062,346 valid samples (null values removed). The measurements were collected between 16/12/2006 and 26/11/2010. Each interval lasts 10 min. We use the data from 16/12/2006 to 24/04/2009 for training, data from 25/02/2010 to 26/11/2010 for testing, and the remaining for validation. RMSE is used as the performance metric.

- Human Activity [3]: This dataset contains smartphone sensor data (accelerometer, gyroscope, and magnetometer) collected from 30 volunteers performing six activities. We predict the activities of volunteers based on the historical data. We use 7,352 instances for training and 2,947 instances for testing following [13]. We use Accuracy (ACC), Precision (P), Recall (R), and F1 as the performance metrics.

2) Baselines and Configurations: We adopt GRU/LSTM as the hyper layers and main layers for our HTSF framework because they suffice for short sequence prediction in many applications [10], [19]. The corresponding models are termed HyperGRU and HyperLSTM. We compare them with widely used baselines, including: LSTM [34], GRU [37], MMD-RNN [38], DANN-RNN [39], LSTNet [8], and ADARNN [13].

These baselines are chosen for the following reasons. LSTM and GRU are basic models in our HyperLSTM and HyperGRU. LightGBM is a popular and strong traditional machine learning algorithm in Kaggle competitions. LSTNet and STRIPE are state-of-the-art RNN-based methods without considering the problem of distribution shift. MMR-RNN, DANN-RNN, and ADARNN are recent models attempting to solve the distribution shift problem in time series, where ADARNN is state-of-the-art.

For all datasets, we use Adam [40] with weight decay regularization as the optimizer. The dataset-specific configurations are summarized below.

- Air Quality: We use L2 loss as the training objective. The learning rate is 2e-4 with a weight decay of 0.01, and the batch size is 128. We set sliding window size \( T = 672 \) (28 consecutive days), \( k = 24 \) for 1D average pooling.

- Electricity Consumption: We use L2 loss as the training objective and 5e-5 as the learning rate. We set sliding window size \( T = 1008 \) (one week), \( k = 36 \) for 1D average pooling, and the batch size is set to 64.

- Human Activity: We use cross-entropy as the training objective and the learning rate is 2e-5 with a weight decay of 0.01. We set sliding window size \( T = 128 \), \( k = 36 \) for 1D average pooling, and the batch size is set to 256.

We set the prediction step \( T_F = 1 \) for all benchmarks as in [13].

3) Results: The left part of Table I shows the RMSE and MAE of different methods for the air quality prediction tasks. The right part of Table I presents the RMSE of predicting the power consumption. Table II lists the ACC, P, R, and F1 scores of different methods of activity classification. Since LSTNet and STRIPE are built for regression tasks, they are not applicable to this dataset. We make the following observations.

- HyperLSTM outperforms LSTNet and STRIPE, the methods that ignore distribution shift, by decreasing the RMSE/MAE by 61.4%/80.6% and 23.9% /34.4% in average. This indicates the importance of addressing the distribution shift problem.

- HyperLSTM and HyperGRU outperform MMD-RNN and DANN-RNN on all benchmarks, suggesting that HTSF better handles distribution shift than existing domain adaptation and generalization methods for time series forecasting.

- Compared to ADARNN, one state-of-the-art method for handling distribution shift, HyperLSTM and HyperGRU decrease the RMSE by 3.94% and 3.82% averaged on the air quality dataset; and 2.60% and 2.60% on the electricity consumption dataset. HyperGRU also outperforms ADARNN by increasing ACC by 1.32% and F1 score by 1.04% on the human activity dataset. These results indicate that HTSF can better characterize the distribution shifts in data.

B. Spatiotemporal Forecasting

Since graph neural network (GNN) based models have achieved the state-of-the-art performance in urban traffic forecasting [19], [20], we are interested in whether HTSF can further improve the performance of GNN-based models.

1) Datasets and Metrics: We conduct experiments on PeMSD3, PeMSD4, PeMSD7, and PeMSD8, which are graph-based datasets collected by the Caltrans Performance Measurement System [41] for traffic flow prediction. All the datasets are split with a ratio of 6:2:2 for training, validation, and testing. The traffic flows are aggregated into 5-minute interval.
We use the following Table IV summarizes the average results on GNN-based models outperform LSTM and GRU since our HyperLSTM and HyperGRU, despite modeling only 0.080 0.0278 0.0178 0.0155 0.0107 0.093 0.0356 0.0255 0.0214 0.0157 0.0274 0.0203 0.0291 0.0211 Dongsi 0.0510 0.0380 0.0519 0.0475 0.0348 0.0459 0.0330 0.0347 0.0544 0.0651 0.0519 0.0651 0.0548 0.0696 0.0599 0.0705 RMSE 0.0138 0.075 0.0502 0.0403 0.0511 0.0459 0.0330 0.0347 0.0189 0.0120 0.0231 unknown 0.080 0.0134 0.0184 0.0119 0.0229 0.0360 0.0267 0.0183 0.0133 0.0267 0.0197 0.0288 0.0168 0.077 Electric Power 0.0365 0.0216 0.0204 0.0148 0.0248 0.0154 0.0304 0.0139 0.082 0.0285 0.0182 0.0156 0.0111 A. C E B L A T I O N x S In this study, we use HyperGRU as benchmarks with its corresponding historical dataset contains 128 sequences, Table V shows comparison results of Hyper- and their original Multi-step prediction capability is crucial for Figure. 3, we observe that the original HyperAGCRN achieves the best performance across all datasets. We set to 3e-4 with a weight decay of 0.01. The learning rate is set to 3e-4 with a weight decay of 0.01. Table III summarizes the detailed information of the datasets. RMSE, MAE, and MAPE are used as performance metrics. 2) Baselines and Configurations: We use the following baselines: DCRNN [42], STGCN [4], ASTGCN [43], GraphWaveNet [44], AGCRN [19] and STFGNN [20]. The selected GNN-based models include representative ( [4], [42]) and state-of-the-art ([19], [20]) models for traffic forecasting. We compare these methods with our HTSF using pure RNN-based models (HyperLSTM and HyperGRU) and an extension of HTSF to AGCRN, a GNN-based model. Specifically, we adopt AGCRN as the hyper layers and main layers to HTSF, and the corresponding model is termed HyperAGCRN. For all four datasets, we use the data of past 12 steps to predict the first 12 steps. We set sliding window size $T = 2016$ (one week), and $k = 24$ for 1D average pooling. We optimize all the models by Adam with batch size 128. The learning rate is set to 3e-4 with a weight decay of 0.01. 3) Results: Table IV summarizes the average results on graph-based spatiotemporal forecasting. We have the following observations. • GNN-based models outperform LSTM and GRU since GNN-based models take advantage of modeling spatial patterns while LSTM and GRU only model the temporal patterns. • Our HyperLSTM and HyperGRU, despite modeling only the temporal patterns, outperform multiple GNN-based models. For example, HyperLSTM and HyperLSTM have better RMSE, MAE, and MAPE than GraphWaveNet on PeMSD3, PeMSD4, and PeMSD7. The reason might be that the temporal distribution shift may be more important than the spatial patterns in the accuracy of traffic prediction. • HyperAGCRN achieves the best performance across all datasets. This demonstrates that HTSF can improve the performance of GNN-based models.

VI. ABLATION STUDIES

A. Visual Analysis of Distribution Shift

1) Setups: In this study, we demonstrate the problem of distribution shift in time series via visual analysis of the Air Quality dataset. For each station, we randomly sample one instance $x^{(i)}$ with its corresponding historical dataset $S = \{ x_n^{(i)} \}_{n=1}^{L}$, where $S$ contains 128 sequences, i.e., $L = 128$. We use t-SNE [45] to visualize the learned distributions (via the hyper layers) of all sequences in $S$ and their original distributions.

2) Results: From Figure. 3, we observe that the original distributions are random while the learned distributions can be approximately fitted by a linear model, indicating that common knowledge among all historical sequences has been learned by HTSF.

B. Impact of 1D Average Pooling

1) Setups: In this study, we use HyperGRU as benchmarks to evaluate the effects of 1D average pooling. We conduct experiments on the PeMSD4 and PeMSD8.

2) Results: Table V shows comparison results of HyperGRU and HyperGRU*. HyperGRU* i.e., without 1D averaging pooling achieves comparable forecasting accuracy as HyperGRU (with 1D averaging pooling). On PeMSD8, HyperGRU achieves even better results on some metrics (RMSE and MAPE).

C. Impact of Prediction Horizon

1) Setups: Multi-step prediction capability is crucial for traffic flow forecasting. We analyze the performance of HyperGRU for multi-step prediction on PeMSD4 and PeMSD8 datasets.

TABLE I PERFORMANCE OF AIR QUALITY AND ELECTRICITY CONSUMPTION.

| Methods      | Dongsi     | Tiantan    | Nongzhanguan | Dingling     | Electric Power |
|--------------|------------|------------|--------------|--------------|----------------|
|              | RMSE       | MAE        | RMSE         | MAE          | RMSE           |
| LSTM         | 0.0302     | 0.0403     | 0.0311       | 0.0453       | 0.0341         |
| GRU          | 0.0510     | 0.0380     | 0.0519       | 0.0475       | 0.0348         |
| LSTMNet      | 0.0544     | 0.0651     | 0.0519       | 0.0651       | 0.0548         |
| MMD-RNN      | 0.0360     | 0.0267     | 0.0183       | 0.0133       | 0.0267         |
| DANN-RNN     | 0.0356     | 0.0255     | 0.0214       | 0.0157       | 0.0274         |
| STRIEPE      | 0.0365     | 0.0216     | 0.0204       | 0.0148       | 0.0248         |
| ADARNN       | 0.0295     | 0.0185     | 0.0164       | 0.0112       | 0.0196         |
| HyperLSTM    | 0.0278     | 0.0178     | 0.0155       | 0.0107       | 0.0189         |
| HyperGRU     | 0.0285     | 0.0182     | 0.0156       | 0.0111       | 0.0184         |

TABLE II PERFORMANCE OF HUMAN ACTIVITY PREDICTION

| Methods      | ACC        | $P$        | $R$        | $F1$        |
|--------------|------------|------------|------------|-------------|
| LightGBM     | 84.11      | 83.73      | 83.63      | 84.91       |
| GRU          | 85.68      | 85.62      | 85.51      | 85.46       |
| MMD-RNN      | 86.39      | 86.80      | 86.26      | 86.38       |
| DANN-RNN     | 85.88      | 85.59      | 85.62      | 85.56       |
| HyperGRU     | 89.61      | 89.64      | 89.45      | 89.43       |

TABLE III DATASET DESCRIPTION OF PEMSD.

| Datasets    | #Detectors | Range       |
|-------------|------------|-------------|
| PeMSD3      | 358        | 9/1/2018-11/30/2018 |
| PeMSD4      | 307        | 1/1/2018-2/28/2018 |
| PeMSD7      | 883        | 5/1/2017-8/31/2017 |
| PeMSD8      | 170        | 7/1/2016-8/31/2016 |
TABLE IV
HTSF ADAPTED FOR GRAPH-BASED TRAFFIC PREDICTION.

| Methods         | PeMSD3 |                  |                  | PeMSD4 |                  |                  | PeMSD7 |                  |                  | PeMSD8 |                  |
|-----------------|--------|------------------|------------------|--------|------------------|------------------|--------|------------------|------------------|--------|------------------|
|                 | RMSE   | MAE              | MAPE             | RMSE   | MAE              | MAPE             | RMSE   | MAE              | MAPE             | RMSE   | MAE              | MAPE             |
| LSTM            | 32.39  | 21.33            | 21.33            | 35.31  | 23.57            | 15.85            | 41.60  | 29.98            | 15.33            | 29.92  | 19.12            | 13.92            |
| GRU             | 32.26  | 21.47            | 21.06            | 38.87  | 24.35            | 15.94            | 41.93  | 30.14            | 15.71            | 30.21  | 19.12            | 13.52            |
| DCRNN           | 30.31  | 18.18            | 18.91            | 38.12  | 24.70            | 17.12            | 38.58  | 28.30            | 11.66            | 27.83  | 17.86            | 11.45            |
| STGCN           | 30.12  | 17.49            | 17.15            | 35.55  | 22.70            | 14.59            | 38.78  | 25.38            | 11.08            | 27.83  | 18.02            | 11.40            |
| ASTGCN          | 29.66  | 17.69            | 19.40            | 35.22  | 22.93            | 16.56            | 42.57  | 28.05            | 13.92            | 28.16  | 18.61            | 13.08            |
| GraphWaveNet    | 32.94  | 19.85            | 19.31            | 39.70  | 25.45            | 17.29            | 42.78  | 26.85            | 12.12            | 31.05  | 19.13            | 12.68            |
| AGRN            | -      | -                | -                | 32.30  | 19.83            | 12.97            | -      | -                | -                | -      | -                | -                |
| STFGNN          | 26.28  | 16.77            | 16.30            | 32.51  | 20.48            | 16.77            | 36.60  | 23.46            | 9.21             | 26.25  | 15.95            | 10.09            |
| HyperLSTM       | 29.32  | 18.91            | 18.06            | 36.19  | 22.65            | 15.46            | 38.67  | 27.18            | 13.17            | 27.12  | 18.10            | 13.68            |
| HyperGRU        | 29.28  | 18.94            | 17.98            | 36.94  | 23.47            | 15.59            | 38.75  | 27.31            | 12.73            | 27.46  | 18.39            | 14.23            |
| HyperAGCRN      | 25.68  | 15.85            | 14.37            | 29.93  | 18.39            | 12.11            | 34.37  | 21.64            | 8.89             | 24.37  | 15.26            | 9.64             |

TABLE V
ABLATION STUDY OF 1D AVERAGE POOLING ON PeMSD4 AND PeMSD8.

| Methods  | PeMSD4 |                  |                  | PeMSD8 |
|----------|--------|------------------|------------------|--------|
|          | RMSE   | MAE              | MAPE             | RMSE   | MAE              | MAPE             |
| LSTM     | 36.19  | 22.65            | 15.46            | 27.46  | 18.39            | 14.23            |
| HyperLSTM| 36.12  | 22.58            | 15.37            | 27.53  | 18.36            | 14.41            |

HyperGRU* removes 1D average pooling from HyperGRU.

2) Results: Figure 4 shows the prediction performance of various methods as the horizon increases. Overall, the errors (in RMSE) grow with the forecast step. However, for both datasets, the errors of AGCRN and HyperAGCRN increase much more slowly than GRU and HyperGRU. HTSF significantly improves the performance of GRU and HyperAGCRN on multi-step prediction, while the aggravation of our HyperAGCRN is much slower than other models. These results indicate that mitigating the distribution shift problem using HTSF is effective in mining the dynamic patterns of spatiotemporal data.

VII. CONCLUSION

In this paper, we investigate the distribution shift problem in the time series forecasting problem, which causes discrepancies between the distributions of the training and the testing data. To this end, we propose HTSF, a novel hypernetwork-based framework which applies for time series forecasting under distribution shift. Specifically, HTSF exploits the hyper layers to learn the best characterization of the distribution shifts, generating the model parameters for the main layers to make accurate predictions. Moreover, HTSF is implemented as an extensible framework that can incorporate diverse time series forecasting models. Extensive experiments show that HTSF outperforms other state-of-the-art methods on 7 benchmarks. One future direction is to introduce parallel support for training and inference, especially the parallel training of hyper layers.

REFERENCES

[1] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, “Cautionary tales on air-quality improvement in beijing,” Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 473, no. 2205, p. 20170457, 2017.
[2] D. Lee and K. Kim, “Recurrent neural network-based hourly prediction of photovoltaic power output using meteorological information,” Energies, vol. 12, no. 2, pp. 1–22, 2019.
[3] B. Almaslukh, J. Aimtadi, and A. Atroli, “An effective deep autoencoder approach for online smartphone-based human activity recognition,” Int. J. Comput. Sci. Netw. Secur., vol. 17, no. 4, pp. 160–165, 2017.
[4] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” in Proceedings of the International Joint Conference on Artificial Intelligence. Burlington, MA, USA: Morgan Kaufmann, 2018, pp. 3634–3640.
[5] J. D. Hamilton, Time series analysis. Princeton, NJ, USA: Princeton university press, 2020.
[6] J. Connor, L. E. Atlas, and D. R. Martin, “Recurrent networks and narma modeling,” in Advances in Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc., 1992, pp. 301–308.
