Modeling Temporal Patterns with Dilated Convolutions for Time Series Forecasting

YANGFAN LI[*], College of Information Science and Engineering, Hunan University
KENLI LI[†], College of Information Science and Engineering, Hunan University
CEN CHEN[*], Institute for Infocomm Research, Singapore
XU ZHOU, College of Information Science and Engineering, Hunan University
ZENG ZENG, Institute for Infocomm Research, Singapore
KEQIN LI, Hunan University State University of New York

Time series forecasting is an important problem across a wide range of domains. Designing accurate and prompt forecasting algorithms is a non-trivial task, as temporal data that arise in real applications often involve both non-linear dynamics and linear dependencies, and always have some mixtures of sequential and periodic patterns, such as daily, weekly repetitions, etc. At this point, however, most recent deep models often use RNNs to capture these temporal patterns, which is hard to parallelize and not fast enough for real-world applications especially when a huge amount of user requests are coming. Recently, CNNs have demonstrated significant advantages for sequence modeling tasks over the de-facto RNNs, while providing high computational efficiency due to the inherent parallelism. In this work, we propose HyDCNN, a novel hybrid framework based on fully Dilated CNN for time series forecasting tasks. The core component in HyDCNN is a proposed hybrid module, in which our proposed position-aware dilated CNNs are utilized to capture the sequential non-linear dynamics and an autoregressive model is leveraged to capture the sequential linear dependencies. To further capture the periodic temporal patterns, a novel hop scheme is introduced in the hybrid module. HyDCNN is then composed of multiple hybrid modules to capture the sequential and periodic patterns. Each of these hybrid modules targets on either the sequential pattern or one kind of periodic patterns. Extensive experiments on five real-world datasets have shown that the proposed HyDCNN is better compared with state-of-the-art baselines and is at least 200% than RNN baselines. The datasets and source code will be published in Github to facilitate more future work.

CCS Concepts: • **Information systems** → **Data mining**; *Information retrieval*; • **Computing methodologies** → *Machine learning*.

Additional Key Words and Phrases: Convolutional neural networks, dilated convolutions, time-series forecasting

[*]Both authors contributed equally to this research.
[†]Corresponding author

Authors' addresses: Yangfan Li, yangfanli@hnu.edu.cn, College of Information Science and Engineering, Hunan University, Lushan Road (S), Yuelu District, Changsha, Hunan, 410082; Kenli Li, lkl@hnu.edu.cn, College of Information Science and Engineering, Hunan University, Lushan Road (S), Yuelu District, Changsha, Hunan, 410082; Cen Chen, chenc@i2r.a-star.edu.sg, Institute for Infocomm Research, Singapore, 1 Fusionopolis Way, 20-10, Connexis North Tower, Singapore; Xu Zhou, zhouxu@hnu.edu.cn, College of Information Science and Engineering, Hunan University, Lushan Road (S), Yuelu District, Changsha, Hunan, 410082; Zeng Zeng, zengz@i2r.a-star.edu.sg, Institute for Infocomm Research, Singapore, 1 Fusionopolis Way, 20-10, Connexis North Tower, Singapore; Keqin Li, likq@hnu.edu.cn, Hunan University, Lushan Road (S), Yuelu District, Changsha, Hunan, 410082, State University of New York, 353 Broadway, Albany, New York.

1556-4681/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Yangfan Li, Kenli Li, Cen Chen, Xu Zhou, Zeng Zeng, and Keqin Li. 2018. Modeling Temporal Patterns with Dilated Convolutions for Time Series Forecasting. *ACM Trans. Knowl. Discov. Data.* 37, 4, Article 111 (August 2018), 22 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Time series data such as traffic flows on highways, outputs of solar power plants, and temperatures of multiple regions are very important to both industry and academia. In these applications, users will benefit from accurate time series forecasting. For example, the energy industry requires accurate forecasting of energy demand, supply and price in their decision-making processes [22], and accurate traffic prediction is critical for many transportation services, such as vehicle flow control, road trip planning and navigation [10]. Moreover, response time is also important in the era of big data [8, 9], e.g., a mobility service provider, such as Uber, has to make tremendous amount of traveling time predictions simultaneously when huge number of user requests are coming. On this occasion, longer waiting time always leads to poorer user experience.

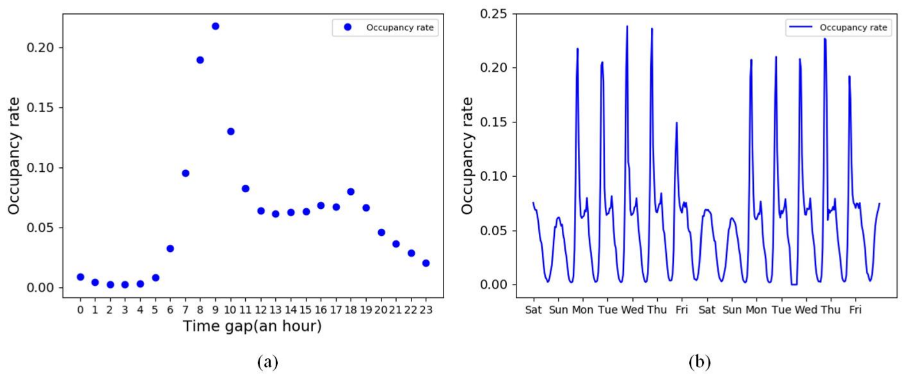
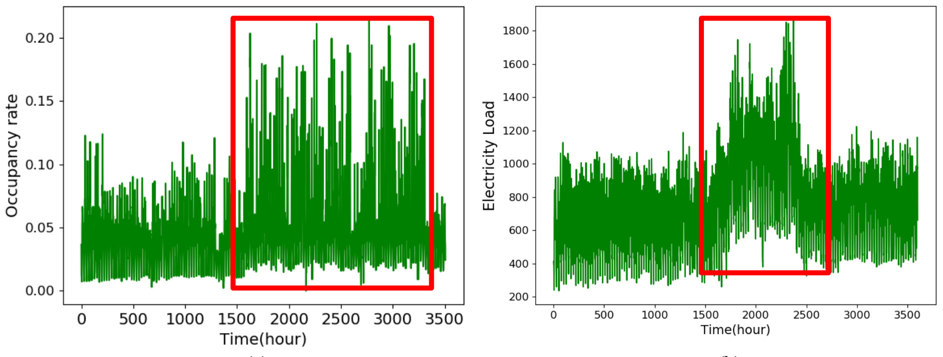


Fig. 1. Illustrations of sequential and periodic patterns.

Real-world time series applications often entail some mixtures of sequential and periodic temporal patterns as well as the linear dependencies and non-linear dynamics. In time series with multi-variables, each variable not only depends on the historical values but also on other variables. Fig. 1(a) presents sequential temporal patterns where the value of interval is affected by the previous ones. For example, traffic jam around 9:00 am will affect the following traffic situation. Fig. 1(b) illustrates several kinds of periodic temporal patterns, such as morning and evening peaks, workday and weekend patterns, etc. Time series data also involve mixtures of linear dependencies and non-linear dynamics [25, 28, 48]. Fig. 2(a) and (b) present hourly road usages and electricity loads in 3500 hours, respectively, both of which have some sudden changes. These changes are often caused by random events, such as traffic jam, holidays, etc. In [49], it has been demonstrated that linear models can capture such situations better than non-linear models. Moreover, there are also rich spatial dependencies among variables. For example, in a road network, each variable signifies the traffic of a road, and the traffic of the road can affect its nearby traffic. Moreover, there are also rich spatial dependencies among time series with multiple variables. For example, in a road network, each variable signifies the traffic of a road, and the traffic of the road can affect its nearby traffic.



(a) (b)

Fig. 2. Sudden changes in scale in time series data.

Previous work [25, 41, 48, 49] combined linear model, such as ARIMA [6], and Multilayer Perceptions (MLP) together for time series prediction, where ARIMA and MLP were utilized for modeling the linear dependencies and non-linear dynamics, respectively. These methods outperformed the models that captured either linear dependencies or non-linear dynamics. However, MLP may fail to extract the complex patterns in sequences, especially comparing to the sequencing models based on deep learning, e.g., RNN or its variants.

Long and Short-term Time-series Network (LSTNet) [28] is considered as a reliable method for time series prediction, which combines a traditional autoregressive linear model with the non-linear recurrent neural network (RNN), to improve the robustness [28]. Results on some real-world datasets show that LSTNet outperforms other methods, such as RNN based and traditional statistics based approaches, etc. Nevertheless, LSTNet still relies on RNN to model long-term temporal patterns. Moreover, it only captures the sequential linear dependencies, while leaves the linear dependencies of temporal periodic patterns untouched.

RNNs have to wait for the predecessors completed before next procedures [38], while convolutions can be done in parallel since the same filters are used in each layers [18]. Therefore, long sequential inputs can be processed together and paralleled in CNN-based methods, instead of sequentially repeated in RNNs, to obtain higher processing speed. Recent research indicates that CNN-based architecture can achieve comparable accuracy in some sequence modeling tasks, such as audio synthesis, word-level language modeling, and machine translation [2, 4, 15, 18, 26]. However, modeling the temporal patterns for time series prediction is still a non-trivial task. We summarize 4 main challenges as follows:

- Capturing the long-term and periodic patterns is infeasible for canonical CNN as the convolution operations are performed within sequential elements in time series data.
- Canonical CNN lacks the ability to learn the temporal positional information because all the records are treated the same by a same set of convolution kernels.
- How to capture the spatial dependencies among variables in multivariate time series data using CNN?
- Although CNNs are quite powerful in capturing non-linear and regular patterns, there are also some random and sudden scale changes shown in Fig. 2 which we experimentally find CNNs are insensitive to.

To solve the first three challenges, we propose the position-aware dilated CNN. 1) The periodical information in time series data are modeled with dilated convolution. This idea is based on a intuition that the dilated convolution operation is performed on the skipped elements rather than consecutive elements. Therefore, a convolution kernel can capture the patterns within the periodic historical records. By changing the dilated factor, different periodical patterns (e.g., daily, weekly) can be captured. 2) To endow the CNNs with the positional sense, we propose the multi-span temporal feature aggregation scheme to enable the convolutional component aware of the temporal features of different time span, and the position embedding scheme to give the model a sense of which part of the entire input series it is dealing with. 3) We let each variable corresponds to one element in the input channel. We then apply 1D CNN to extract the channel-wise correlations along with the temporal correlations so that the spatial dependencies among variables can be captured, which is a simple yet effective method. For the 4-th challenge, we analyze that the scale change can be modeled by the sequential and periodic linear dependencies, i.e., the future value is assumed to be a linear combination of some past sequential and periodic values with random errors. Incorporating linear models such as AR is a robust solution in modeling linear dependencies.

In order to exploit both non-linear dynamics and linear dependencies for capturing sequential and periodic patterns, we propose the hybrid framework HyDCNN which contains multiple hybrid modules with a novel hop scheme. The tasks of capturing sequential and periodic patterns are decoupled inter-module, while the objectives of modeling non-linear dynamics and the linear dependencies are achieved by the intra-module position-aware dilated CNN and AR components respectively. Each hybrid module targets on either the sequential pattern or one kind of periodic patterns base on it hop factor. HyDCNN is an end-to-end trainable framework and can fuse multiple hybrid modules adaptively. Extensive experiments on five real-world datasets have shown that the proposed HyDCNN is better compared with state-of-the-art baselines and is at least 200% than RNN baselines.

The remaining of the work is organized as follows. In the next section, the related work is discussed. Section 3 describes our problem definition and the overview of HyDCNN. Section 4 presents the details of HyDCNN. Section 5 presents the experimental results, while the work is concluded in Section 6.

2 RELATED WORK

Approaches for the prediction of time series data can be mainly classified into three categories: traditional statistics based, artificial neural networks based, and hybrid approaches, as discussed in the following.

2.1 Traditional Statistics based Approaches

Traditional statistics based approaches for time series forecasting can be further categorized into two sub-classes: linear and non-linear based models. Auto-Regression (AR), Vector Auto-Regression (VAR) [6, 34], Moving Average (MA), Auto-Regressive Moving Average (ARMA), Auto-Regressive Integrated Moving Average (ARIMA) model [6], and regression based methods such as linear Support Vector Regression (SVR) [7, 27] are linear models where prediction of time series values is constrained to be linear functions of past values. To account for certain non-linear patterns observed in real problems, several classes of non-linear models have been proposed, such as the bilinear model [19], and Auto-Regressive Conditional Heteroscedastic (ARCH) model [17]. Although some improvement has been achieved by these non-linear models, the gain of using them to general forecasting problems is still limited. Because these models are designed for some particular non-linear patterns, they fail to model general ones in time series data [16].

2.2 Artificial Neural Networks based Approaches

Artificial Neural Networks (ANNs), especially Deep Neural Networks, have been proposed for time series forecasting recently due to their powerful ability of feature learning and unprecedented successes in other domains. DeepAR [39] proposed an autoregressive recurrent neural network model on a large number of related time series. Romeu et al. [37] present an approach to probabilistic time series forecasting that combines state space models with deep learning. N-beats [36] present an approach to probabilistic time series forecasting that combines state space models with deep learning. Another application in [35] adopted Stacked Autoencoders to predict the future traffic flow based on historical traffic flow values. However, sequential and periodic patterns of time series data were not taken into account in the work.

Similarity based methods are powerful in time series analysis [1]. Wu et al. [44] proposed a family of alignment-aware positive definite kernels, with its feature embedding given by a distribution of Random Warping Series. Lei et al. [29] proposed an efficient representation learning framework that is able to convert a set of time series with various lengths to an instance-feature matrix. Cuturi [13] proposed novel approaches to cast the widely-used family of Dynamic Time Warping (DTW) distances and similarities as positive definite kernels for time series. However, sequential and periodic patterns of time series data were not taken into account in the work.

Recently, Recurrent Neural Networks (RNNs) [38, 43] and its variants, e.g., Long Short-Term Memory (LSTM) [21], Gated Recurrent Unit (GRU) [12] have achieved great success in sequential tasks, such as Natural Language Processing (NLP). Models based on RNNs for time series prediction have also been proposed. TreNet [31] combined CNNs and LSTM to predict the trend in time series. But RNNs may failed on tasks that require long-term information, mainly due to the notorious gradient vanishing problem [12] and are hard to parrallelize.

Researchers have also investigated how to use deep learning to capture the spatial dependencies among variables in time series with multi-variables. DCRNN [30] and [45] use graph to model the relation between variables. The nodes in the graph represent the variables and the edges describe the relations of variables. DSANet [24] first extracts the features from each unvariate time series and then apply a self-attention module to learn the dependencies among variables. Comparing with them, our HyDCNN mainly focus on capturing the temporal patterns. Actually, we can also capture the spatial dependencies among vectors. We use a simple yet also effective scheme to model these spatial dependencies. In the first layer of CNN, we assign each variable to one input channel. The convolution operation will capture the channel-wise correlations therefore the spatial dependencies are also captured.

Recent works also use CNNs for time series prediction. SOCNN [3] involved an autoregressive-like weighting system in CNNs to deal with the noise in time series data. G-CNN [46] used group convolutions for high-dimensional multivariate regression tasks. TCN [2] and [5] apply dilated convolution for time series forcasting. But None of them considers the problem of how to use CNN to capture the long-term periodic patterns in time series data and how to capture the linear dependencies.

2.3 Hybrid Approaches

Some research work has explored hybrid methods for time-series prediction. ARIMA and MLP were combined together for time series prediction in [25, 48, 49]. They achieved some advantages compared with the methods which utilized either ARIMA or MLP only. These work utilized ARIMA to capture the linear dependencies and MLP to model the non-linear dynamics. However, it is hard for the work to model sequential dependencies.

A recent work was proposed by Dasgupta et al. [14] that combined vanilla RNN and Dynamic Boltzmann Machines together. Though this work can utilized RNN to model the sequential dependencies, the long-term periodic temporal patterns cannot be exploited. To tackle this, LSTNet was proposed in [28] for time series prediction and outperformed other hybrid and non-hybrid methods. In LSTNet, a novel recurrent structure, namely Recurrent-skip, was designed based on RNN for capturing long-term imformation. However, it only captured the sequential linear dependencies, while left the linear dependencies of periodic temporal patterns unexploited.

Compared with the discussed methods, our proposed method has the following advantages: (1) HyDCNN is based on CNNs to capture both the sequential and periodic dependencies, thus owning the advantages of computations due to the inherent parallelism of CNNs. (2) HyDCNN can capture both the linear dependencies and non-linear dynamics for both sequential and periodic patterns.

3 PROBLEM DEFINITION AND FRAMEWORK OVERVIEW

In the section, we present the definition of time series forecasting and overview of the proposed framework as follows.

3.1 Problem Definition

In this work, we focus on the problem of time series forecasting. Similar with [28], given a series of historical values $\boldsymbol{X}_T = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_T\}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$, n is the number of multiple variables, and T is the length of the entire series. Obviously, when $n = 1$, the problem is a univariate forecasting problem or multivariate forecasting problem otherwise. Let w be the length of the input window where w is integer and $1 < w \leqslant T$ holds. Then the input matrix for sequential time series values at time t can be formulated as $\boldsymbol{X}_t = \{\boldsymbol{x}_{t-w+1}, \boldsymbol{x}_{t-w+2}, ..., \boldsymbol{x}_t\}$ and $\boldsymbol{X}_t \in \mathbb{R}^{n \times w}$. The time series forecasting problem is to predict a series of values $\boldsymbol{x}_{t+h} \in \mathbb{R}^n$, where h is the desirable horizon ahead of t. In most cases, the horizon h and the input window w are set according to the demands of real scenarios, e.g., for the traffic usage, the horizon of interest ranges from hours to a day and the window ranges from days to weeks.

3.2 Framework Overview

Sequential and periodic patterns are both crucial for accurate prediction. Exploiting both the non-linear dynamics and linear dependencies is an effective way to capture these patterns. To achieve this goal, we propose the HyDCNN shown in Fig. 3, where the inputs of HyDCNN are the historical time series values, while the outputs are the predictions of future time signals. HyDCNN contains multiple hybrid modules, and the tasks of capturing sequential and periodic patterns are decoupled inter-module, while the objectives of modeling non-linear dynamics and linear parts are achieved by the intra-module CNN and AR components respectively.

HyDCNN consists of a hybrid module, which targets on the sequential temporal patterns, and multiple hop hybrid modules, each of which captures one kind of periodic temporal pattern, e.g., daily, weekly, etc. In each module, there is a position-aware dilated CNN component to model the non-linear dynamics, and a hop or non-hop AR component to model the linear parts. These two components are combined by a residual learning method. The features extracted by the hybrid module and hop hybrid modules are finally fused together by our proposed weighted fusion scheme to represent the sequential and multiple periodic temporal patterns jointly. For the sake of simplicity, there is only one hop hybrid module in Fig. 3, and more hop hybrid modules can be combined into the framework in the same way to capture multiple temporal patterns.

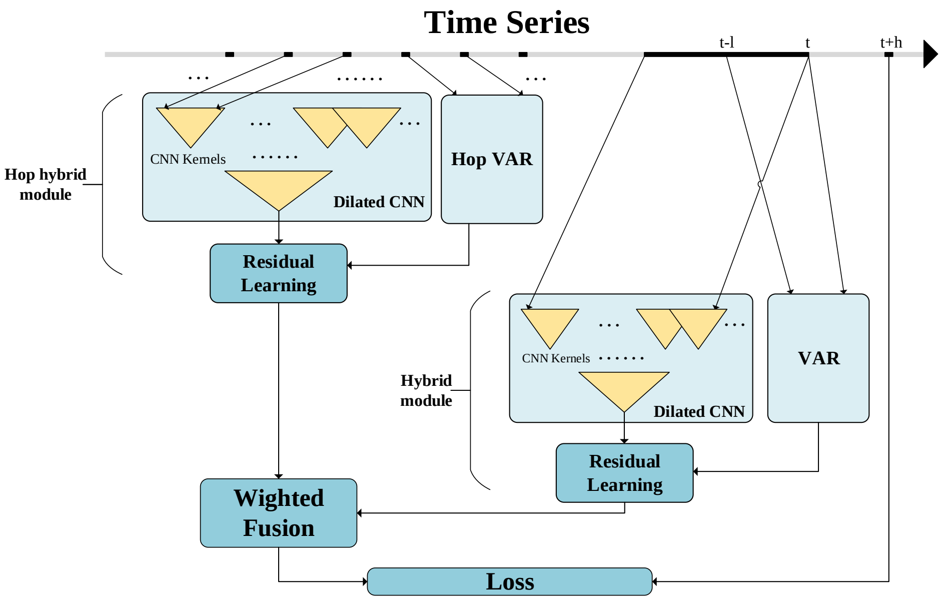


Fig. 3. Architecture of our proposed HyDCNN

4 THE PROPOSED FRAMEWORK: HYDCNN

In this section, we present the details of HyDCNN. First, we describe the original hybrid module for sequential temporal patterns. Second, we introduce how to adopt the hop scheme in the original hybrid module to capture periodic temporal patterns. Finally, we introduce how to fuse multiple hybrid modules, the objective function and the optimization strategy together.

4.1 Hybrid module for Sequential Temporal Patterns

In HyDCNN, a hybrid module consists of a position-aware dilated CNN stack for sequential non-linearity, and an AR model for the sequential linear dependencies. These two parts are combined by using residual learning, where the CNN based stack is fitting the residual errors of the outputs of both the hybrid model and the AR model. The input of the original hybrid module is historical time series values. Take time t as an example, the input matrix can be formulated as $\boldsymbol{X}_t = \{\boldsymbol{x}_{t-w+1}, \boldsymbol{x}_{t-w+2}, ..., \boldsymbol{x}_t\}$ and $\boldsymbol{X}_t \in \mathbb{R}^{n \times w}$.

4.1.1 Position-Aware Dilated CNN. Unlike RNNs which have the intrinsic temporal sense by learning the position information through recurrent hidden state computation (the more previous cell has lower gradient) , CNN based models lack this temporal sense because all the records in the time series are treated the same by a same set of convolution kernels [18, 42]. To tackle this problem, we propose the position embedding scheme to give the model a sense of which part of the entire input series it is dealing with and the multi-span temporal feature aggregation scheme to enable the convolutional component aware of the temporal features of different time span. Our proposed position-aware dilated CNN is based on the dilated 1D CNN network [33] shown in Fig. 4 with details as follows.

Position Embedding. CNN based models lack the ability to learn position information, while RNNs can learn the position information through recurrent hidden state computation thus know where they are [18, 42]. To tackle this problem, position embedding is employed to give the model a sense of which part of the entire input series it is dealing with.

Formally, given an input time series $\boldsymbol{X}_t = \{\boldsymbol{x}_{t-w+1}, \boldsymbol{x}_{t-w+2}, ..., \boldsymbol{x}_t\}$, we first use one-hot encoding to represent each position. For example, if the length of a series is 3, the one-hot vectors of the positions are denoted $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$ respectively. After one-hot encoding, we have $\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_w\}$ where $\boldsymbol{e}_i \in \mathbb{R}^w$ denotes the one-hot representation of position i. But these one-hot vectors are high-dimensional sparse vectors are less representative for neural networks to process. To tackle this, we introduce an embedding layer to transform them into low dimensional dense vectors. Formally, the embedding operation $\psi : \mathbb{R}^w \longrightarrow \mathbb{R}^m$ is a linear transformation that transforms the sparse vector $\boldsymbol{e}_i \in \mathbb{R}^w$ into a dense vector $\boldsymbol{p}_i \in \mathbb{R}^m$. The embedding transformation is formulated as follows:

$$\boldsymbol{p}_i = \psi(\boldsymbol{e}_i) = \boldsymbol{e}_i^{\mathrm{T}} \boldsymbol{W}_{\boldsymbol{p}}, \tag{1}$$

where $\boldsymbol{W}_{\boldsymbol{p}} \in \mathbb{R}^{w \times m}$ is a learnable parameter matrix and i denotes the i-th position. Then we have position matrix $\boldsymbol{P}_t = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_w\}$, where $\boldsymbol{p}_i \in \mathbb{R}^m$ denotes the embedding vector of the i-th $(1 \leqslant i \leqslant w)$ absolute position and m is dimension of the embedding vector. For instance, the first absolute position is embedded into $\boldsymbol{p}_1$ and the u-th absolute position is embedded into $\boldsymbol{p}_u$.

Some previous studies have employed position embedding for NLP [15, 18]. The word vectors and the position vectors are added together to form a new representation of words. However, for time series problem, adding up the two embedding matrix is not applicable because each dimension of the time series $\boldsymbol{X}_t$ is a meaningful unvariate time series. If we add the time series values with position vectors together, the actual time series values will be changed.

To address this problem, we concatenate the input matrix with the embedded position matrix as the input of the dilated CNNs, defined in the following.

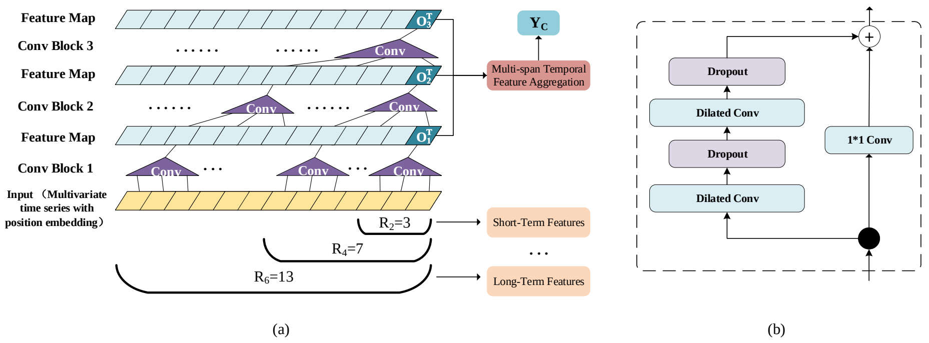


Fig. 4. Position-aware Dilated CNN. (a) An example of position-aware dilated convolutional operation. (b) Block structure and residual connection.

Definition 1 (Position Embedding): Formally, given an input sequence of time series values, $\boldsymbol{X}_t = \{\boldsymbol{x}_{t-w+1}, \boldsymbol{x}_{t-w+2}, ..., \boldsymbol{x}_t\}$, after position embedding, the absolute position of each value is embedded into a sequence of position values, $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_w\}$. The input matrix to the dilated CNNs is $\boldsymbol{C}_t = \{\boldsymbol{c}_{t-w+1}, ..., \boldsymbol{c}_t\} = \{\boldsymbol{x}_{t-w+1} \circ \boldsymbol{p}_1, ..., \boldsymbol{x}_t \circ \boldsymbol{p}_w\}$, where $\circ$ is the concatenation operation.

Fully Convolutional Architecture. The fully convolutional architecture is shown in Fig. 4(a). The input of the position-aware dilated CNN is the sequence of time series values together with the position values.

Formally, given a sequence input $\boldsymbol{C}_t = \{\boldsymbol{c}_{t-w+1}, ..., \boldsymbol{c}_t\}$ where $\boldsymbol{c}_i \in \mathbb{R}^{n+m}$ and a kernel with weight $\boldsymbol{f} \in \mathbb{R}^{k \times (m+n)}$, the position-aware dilated convolution operation F on the element with the index s in the sequence is defined as follows:

$$F(\boldsymbol{C}, s, d) = \sum_{i=0}^{k-1} \boldsymbol{f}_i^{\mathrm{T}} \boldsymbol{c}_{s-d\cdot i}, \tag{2}$$

where d is the dilation factor, k is the kernel size, $s - d \cdot i$ accounts for the direction of the past. When $d = 1$, F becomes a regular convolution. Then, we perform the convolution F on the input sequence $\boldsymbol{C}_t$ to generate one feature map denoted as $\boldsymbol{C}'_t \in \mathbb{R}^w$ as follows:

$$\boldsymbol{C}'_t = \{c'_{t-w+1}, c'_{t-w+2}, ..., c_t\} = \{F(\boldsymbol{C}, t-w+1, d), F(\boldsymbol{C}, t-w+2, d), ..., F(\boldsymbol{C}, t, d)\}. \tag{3}$$

Finally, we apply multiple kernels formulated by Equ. 2 and Equ. 3 to extract different information. A position-aware dilated residual CNN is then re-constructed by stacking multiple position-aware dilated convolutional layers. The output of the r-th layer is the intput of the $(r+1)$-th layer. Two layers with the same dilation size form a block is shown in Fig. 4(b). Several these CNN-based blocks are stacked to form the hierarchical architecture. The final output is denoted as $\boldsymbol{Y}_c \in \mathbb{R}^n$.

It is worth noting that in an original hybrid module, the dilation factor d of the first CNN layer is set to 1 in order to capture the sequential temporal patterns. The dilation factor d in other higher CNN layers is set to 2 or greater to extend the receptive field. The detailed formulation of the receptive field is presented as follows.

Capturing Spatial Dependencies Among Variables and Position Information. A convolution kernel can extract informative features by fusing both the local (local here refers to the size of the kernel) and the channel-wise features together. With a sets of kernels, the local and cross-channel correlations are mapped as new combination of features. This idea has been widely accepted by the literature [11, 23].

In our formulation, we apply kernel with weight $\boldsymbol{f} \in \mathbb{R}^{k\times(m+n)}$ on the time series with position information. There are $m+n$ channels in kernels which correspond to the number of variables and the position embedding. The kernels move 1-dimensionly over the time dimension by Equ. 3. The cross-channel correlations here, representation the dependencies among variable and the position information, can be extracted and represented as new feature maps denoted as $\boldsymbol{C}'_t$ by the convolution. The prediction for each variable is made upon the new features after several convolution layers which contain rich spatial dependencies among variables as well as the position information.

Residual Connection. If we want to extend the receptive field to leverage more long-term temporal information, more convolutional blocks are required and the network will become deeper. Residual neural network is used with a reference to the direct hidden state, instead of an unreferenced function. The residual learning and linear shortcut connections can help to solve the gradient exploding and vanishing problems in the long-term back-propagation [20], especially in the networks with deeper structures.

With the benefits of residual neural networks [20], we introduce the residual error into our proposed fully convolutional architecture as shown in Fig. 4(b). The residual connection is formulated as follows:

$$\boldsymbol{x}_{i+1} = ReLU(\boldsymbol{x}_i + \phi(\boldsymbol{x}_i)), \tag{4}$$

where ϕ denotes the function of the i-th block, $\boldsymbol{x}_i$ denotes the input of the i-th block and $\boldsymbol{x}_{i+1}$ denotes the output of the i-th block and the input of the $(i+1)$-th block. The rectified linear unit (ReLU) is utilized as the activation function. An additional 1×1 convolution is utilized to ensure that element-wise addition receives tensors of the same shape.

Multi-span Temporal Feature Aggregation. The multi-span temporal feature aggregation scheme further lets the convolutional component aware of the position information by modeling the temporal feature of multiple time span. Motivated by FPN [32], which suggest that all the feature maps of different convolutional layers are semantically strong, we further generate an insight that the feature map of each layer can intrinsically represent temporal patterns during the time interval within the receptive field of this layer. The feature maps at different convolution layer can represent the patterns of different time span. More specifically, the receptive time span R_j of the j-th layer is

$$R = 1 + 2\sum_{i=0}^{N} D_i \cdot (K_i - 1), \tag{5}$$

For example, shown in Fig. 4 illustrates a CNN structure with 6 layers (3 residual blocks), where kernel sizes $K_i = 2$ for all the 6 layers, and dilation factor $\boldsymbol{D} = \{1, 1, 2, 2, 3, 3\}$ for the 6 layers respectively. Each triangle represents a residual convolutional block with 2 convolutional layers. One block can access 3 elements. According to Equation (5), the receptive field $\{R_2, R_4, R_6\}$ turns out to be $\{3, 7, 13\}$. Then, part of the feature maps of the 3 layers, i.e., $\boldsymbol{x}_1^T$, $\boldsymbol{x}_2^T$, and $\boldsymbol{x}_3^T$, can represent the most recent 3, 7, 13 records respectively. In real world settings, the receptive time span of the button layer is often small, signifying the short-term features while the receptive time span of the top layer is large, representing the long-term features. Fusing the 3 latent vectors can represent the multi-span temporal feature at time T, formulated as follows:

$$\boldsymbol{Y}_c = W_1 \cdot \boldsymbol{x}_1^T \circ W_2 \cdot \boldsymbol{x}_2^T \circ \ldots \circ W_N \cdot \boldsymbol{x}_N^T, \tag{6}$$

where $W_i \in \mathbb{R}$ is a trainable weight and $\circ$ is the concatenation operator. We introduce the weight parameter because in real-world scenarios, the degree of influence of different layers may be different.

4.1.2 Autoregressive Model. One major limitation of utilizing CNN for time series forecasting is that, although CNN is quite effective for capturing non-linear dynamics especially regular patterns, CNN is not sensitive to the random and sudden changes in time series due to the non-linear nature of CNN. We analyze that the scale change can be modeled by the sequential and periodic linear dependencies, i.e., the future value is assumed to be a linear combination of some past sequential and periodic values with random errors. For example, to predict the burst traffic jam at 9 o'clock today in the traffic dataset, the scale change of road occupancy at the sequential 6, 7, and 8 o'clock today and at the periodic 9 o'clock in the previous day can be inherited for prediction through linear combination. Therefore, to overcome this drawback of CNN, we employ an autoregression model (AR) in our hybrid module to make our HyDCNN more sensitive and robust to sudden and random changes in time series data. The AR module in the origin hybrid only captures the sequential linear dependencies while the periodic linear dependencies are decoupled to other modules which will be discussed later. The AR model is formulated as follows:

$$\boldsymbol{Y}_{ar,t} = \sum_{k=0}^{q^{ar}-1} \boldsymbol{W}_k^{ar} \otimes \boldsymbol{x}_{t-k} + \boldsymbol{b}^{ar}, \tag{7}$$

where $q^{ar} \in \mathbb{R}$ is the length of the input window of AR, $\boldsymbol{W}^{ar} \in \mathbb{R}^{q^{ar} \times n}$ and $\boldsymbol{b}^{ar} \in \mathbb{R}^n$ are the coefficients of AR model, $\boldsymbol{x}_i \in \mathbb{R}^n$ is the i-th element of the input time series with n variables, $\boldsymbol{Y}_{ar,t} \in \mathbb{R}^n$ is the output of AR model at time t, and $\otimes$ is the Hadamard product. This equation means that we apply the AR model to each variable in the time series.

4.1.3 Combining with Residual Learning. The final output of the hybrid module is the combination of the outputs of the position-aware dilated CNN part and the AR component. Formally, the final output is given as follows:

$$Y_s = Y_c + Y_{ar}, \tag{8}$$

where Y_s is the output of a hybrid module, Y_c and Y_{ar} are the outputs of the dilated CNN and AR model respectively. This indicate that the CNNs try to fit the residual errors of the true value and the output of the AR model. The residual errors contain only non-linear dynamics and is learnt by our proposed position-aware dilated CNN.

4.2 Hop Hybrid Module for Periodic Temporal Patterns

The original hybrid module can be utilized to capture the sequential patterns with linear dependencies and non-linear dynamics. As discussed in Section 1, many real-world time series datasets present clear periodic patterns. For instance, both the electricity consumption and traffic usage exhibit periodic patterns on a daily basis. If we aim to predict the traffic usage at t o'clock, a classical strategy for periodic forecasting is to utilize the records at t o'clock of historical days. This type of dependencies also requires an extremely large receptive field.

To tackle this, we introduce a novel hop mechanism into our proposed hybrid module to leverage the periodic patterns and long-term information in real-world scenarios. There are two components in the proposed hybrid module: AR module and position-aware dilated CNNs. The details of how to apply the hop scheme in both components are described separately in the following.

Hop Scheme in the Position-aware Dilated CNNs. Canonical CNN performs the convolution operations within sequential elements in time series data, making capturing the long-term and periodic patterns infeasible. Designing a convolutional component that can effectively dealing with periodical information in time series data is our first objective.

We address this problem by modeling the periodical information in time series data with dilated convolution. This idea is based on a intuition that the dilated convolution operation is performed on the skipped elements rather than consecutive elements. Therefore, a convolution kernel can capture the patterns within the periodic historical records. By changing the dilated factor, different periodical patterns (e.g., daily, weekly) can be captured. In our proposed hop hybrid module, the dilation factor of the first CNN block is set to the hop factor. Take Fig. 5 as an example, suppose the hop factor is set to 3, then the dilation factor of the first CNN block is also 3. The dilation factor of other higher blocks can be set to that in the original hybrid module. Similarly, we can set the dilation factor of the first CNN block to 24 hour to capture the non-linear dynamics of daily patterns. Accordingly, the position embedding with the hop scheme is formulated in the following.

Definition 2 (Hop Position Embedding): Formally, given an input sequence of time series values with the hop factor hp, $\boldsymbol{X}_t = \{\boldsymbol{x}_{t-(w-1)\times hp}, \boldsymbol{x}_{t-(w-2)\times hp}, ..., \boldsymbol{x}_t\}$, after position embedding, the absolute position of each value is embedded into a sequence of position values, $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_w\}$. The input matrix to the first CNN block is $\boldsymbol{C}_t = \{\boldsymbol{c}_{t-w+1}, ..., \boldsymbol{c}_t\} = \{\boldsymbol{x}_{t-(w-1)\times hp} \circ \boldsymbol{p}_1, ..., \boldsymbol{x}_t \circ \boldsymbol{p}_w\}$, where $\circ$ is the concatenate operation.

Hop AR Model for Periodic Linear dependencies. We design a novel hop AR model to capture the periodic linear dependencies. The hop AR model is formulated as follows:

$$Y_{har,t} = \sum_{k=0}^{q^{har}-1} \boldsymbol{W}_k^{har} \otimes \boldsymbol{x}_{t-hp\times(k+1)+h} + \boldsymbol{b}^{har}, \tag{9}$$

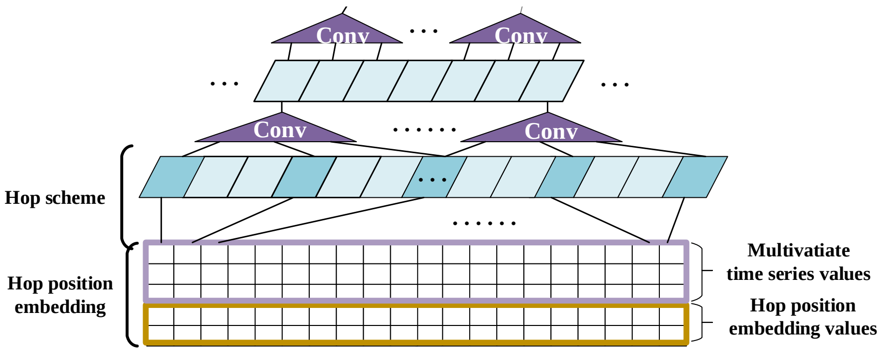

Conv
Conv
Conv
Conv
Hop scheme
Hop position embedding
Multivatiate time series values
Hop position embedding values

Fig. 5. Hop scheme in position-aware dilated CNN.

where hp is the hop factor, $q^{har} \in \mathbb{R}$ is the length of the input window of hop AR, $\boldsymbol{W}^{har} \in \mathbb{R}^{q^{har} \times n}$ and $\boldsymbol{b}^{har} \in \mathbb{R}^n$ are the coefficients of AR model, $\boldsymbol{x}_i \in \mathbb{R}^n$ is the i-th element of the input series, h is the horizon, and Y_{har} is the output of hop AR model.

Take a daily periodic hybrid module as an example: Given time series values before 8, if we want to predict the time series values at 11 o'clock, we have the horizon is 3. It is reasonable for the hop AR Model to take the values of yesterday 11 o'clock, the values of 11 o'clock the day before yesterday and so on as inputs to capture the linear dependencies of daily patterns.

4.3 Fusion Methods for Multiple Hybrid Modules

As discussed in Section 1, many real-world time series datasets present multiple temporal patterns, and the degrees of influence may be different. Inspired by this observations, we propose a novel parametric-tensor-based fusion method that can fuse the outputs of multiple hybrid modules. The equation of fusion method is defined as follows:

$$Y_f = Y_s + \sum_{i=1}^{\gamma} W_i \otimes Y_p^i, \tag{10}$$

where Y_f denote the fused features, $\otimes$ is Hadamard product (i.e., element-wise multiplication for tensors), Y_s are the output features of the hybrid module for sequential patterns, γ denotes the number of hop hybrid modules, Y_p^i means the output features of the hop hybrid module with the index i, and $W_1, ..., W_\gamma$ are the trainable parameters that can adjust the impacts of different hop modules.

We adopt this formulation because sequential patterns are crucial and the periodic patterns are auxiliary in real-world scenarios. The weights of the periodic patterns vary a lot in different time series tasks. Therefore, we set the impacts of different periodic patterns as trainable weights.

4.4 Loss Function

In the training process, we adopt mean squared error (L2-loss) as the objective function, the corresponding optimization objective is formulated as follows:

$$\min_{\Theta} \sum_{j=0}^{M} \sum_{i=0}^{G} (\boldsymbol{x}_{t+h,i}^{j} - \hat{\boldsymbol{x}}_{t+h,i}^{j})^2, \tag{11}$$

Table 1. Dataset details, where LE is the number of time series values, NU is the dimension of the variables, SR is the sample rate.

Datasets	LE	NU	SR
SML2010	4111	1	15 minutes
GEFCom2014 Electricity Load	21552	1	1 hour
Traffic	17544	862	1 hour
Solar-Energy	52560	137	10 minutes
Electricity	26304	321	1 hour

where Θ denotes the parameter set of our model, and h is the user defined horizon, and n is the dimension of data, and G is the number of training samples.

5 EXPERIMENTS

5.1 Experimental Settings

We conducted extensive experiments with seven methods (including HyDCNN) on five real-world datasets for both multivariate and univariate time series forecasting tasks.

5.1.1 Dataset Description. We use five real-world benchmark datasets, including three typical mutivariate datasets [28]: Traffic, Solar Energy, Electricity and two univariate datasets: SML2010 and GEFCom(2014) Electricity Load [22]. Table 1 illustrates the corpus statistics. All the datasets are publicly available.

Multivariate.

- Traffic: A collection of 48 month (2015-2016) hourly data from California Department of Transportation. The data describe the road occupancy rates (between 0 and 1).
- Electricity: The hourly electricity consumption in kWh from 2012 to 2014, for $n = 321$ clients.
- Solar-Energy: The solar power production records in the year 2006, which is sampled every 10 minutes from 137 PV plants in Alabama State.

Unvariate.

- SML2010: The dataset is a uci open dataset [47] used for temperature prediction. This dataset is collected from a monitor system mounted in a domotic house. It corresponds to approximately 40 days of monitoring data.
- GEFCom2014 Electricity Load (GEFCom): The dataset was published in GEFCom2014 forecasting competition and describes the total electricity load of an entire zone.

In our experiments, all datasets have been split into training set (60%), validation set (20%) and test set (20%) in chronological order.

5.1.2 Methods for Comparison. We compare HyDCNN model with the following methods.

- AR stands for the autoregressive model, which is equivalent to 1D VAR model.
- VAR-MLP is the model proposed in [49] that combines Multilayer Perception (MLP) and autoregressive model.
- GRU is a simple, yet powerful variant of RNNs with gating mechanism for time series prediction [12].
- TCN is a general powerful pure CNN-based model for sequence modeling [2].
- LSTNet is a method proposed by [28] that can capture long-term and short-term temporal patterns.

- TPA-LSTM [40] proposes an attention-based model for time series forecasting.
- MTGNN [45] designs a graph neural network framework for time series forecasting.
- HyDCNN is our proposed hybrid model. The input is multivariate time series.

AR is are classical baselines, while GRU is a variant of RNN-based methods. To our best knowledge, MTGNN is state-of-the-art method based on deep neural networks for mutivariate time series forecasting.

5.1.3 Evaluation Metrics. Following LSTNet [28], we use conventional evaluation metrics. All the metrics are commonly used in the corresponding tasks as defined in the following.

- Root Relative Squared Error (RRSE):

$$RRSE = \frac{\sqrt{\sum_{(i,t)\in\Omega_{Test}}(Y_{it} - \hat{Y}_{it})^2}}{\sqrt{\sum_{(i,t)\in\Omega_{Test}}(Y_{it} - \bar{Y})^2}}, \tag{12}$$

- Relative Absolute Error (RAE):

$$RAE = \frac{\sum_{(i,t)\in\Omega_{Test}} \left|Y_{it} - \hat{Y}_{it}\right|}{\sum_{(i,t)\in\Omega_{Test}} \left|Y_{it} - \bar{Y}\right|}, \tag{13}$$

- Empirical Correlation Coefficient (CORR):

$$CORR = \frac{1}{n}\sum_{i=1}^{n} \frac{\sum_t (Y_{it} - \bar{Y}_i)(\hat{Y}_{it} - \bar{\hat{Y}}_i)}{\sqrt{\sum_t (Y_{it} - \bar{Y}_i)^2 (\hat{Y}_{it} - \bar{\hat{Y}}_i))^2}}, \tag{14}$$

where Y is the ground truth, $\bar{Y}$ is the mean, and $\hat{Y}$ is the estimation.

5.2 Implementation Details

PyTorch 1.0.1 is used to build our models [1]. We train and evaluate all the methods on a server with 4 Tesla P100 GPU and 8 E5-2620V4 CPU.

In our implementation, two hybrid modules are combined together in HyDCNN. One original hybrid module is utilized to capture the sequential temporal patterns and one hop hybrid module is utilized to capture the periodic patterns. In the hybrid module, we stack 3 blocks. The dilation factor is set as $\boldsymbol{D} = \{D_1, D_2, D_3\} = \{1, 2, 3\}$, where D_i denotes the dilation factor of the i-th block. In the hop hybrid module, two blocks are stacked. The hop hybrid module is utilized to capture the daily temporal patterns. The hop factor of hop hybrid module is selected based on the sampling frequency of data series, i.e., $hp = 24 \times r$ where 24 means 24 hours for daily periodic patterns and r denotes the hourly sampling rate. For example, for the hourly sampled datasets (GEFCom, Trafc, and Electricity), the hop factor is set to 24. For SML2010 dataset, the sampling rate is 4 per hour and the hop factor is set to 96, while for Solar-Energy dataset, the sampling rate is 6 per hour and that is set to 144.

For other hyper-parameters in HyDCNN, we conduct the grid search scheme. Specifically, the number of kernels is chosen from $\{40, 50, ..., 100\}$ for multivariate datasets and $\{10, 12, ..., 20\}$ for univariate datasets. The kernel size of each block is chosen from $\{3, 4, ..., 10\}$. The dropout rate is set to 0.1 or 0.2. The learning rate is set to 0.001 or 0.01.

For GRU, the number of layers is set to 4, and the hidden state size is chosen from $\{32, 50, 100\}$ via grid search. For LSTNet, we employ the open-source implementation that also adopts the

[1] The source code of HyDCNN can be provided upon request and will be released after review together with the dataset.

Table 2. Experimental results summary of all methods on multivariate datasets (in RSE, RAE and CORR metrics). Best results are displayed in **boldface**.

dataset		Traffic				Solar-Energy				Electricity			
		horizon				horizon				horizon			
methods	metrics	3	6	12	24	3	6	12	24	3	6	12	24
AR	RRSE	0.5991	0.6218	0.6252	0.6293	0.2435	0.3790	0.5911	0.8699	0.0995	0.1035	0.1050	0.1054
	RAE	0.4491	0.4610	0.4700	0.4696	0.1846	0.3242	0.5637	0.9221	0.0579	0.0598	0.0603	0.0611
	CORR	0.7752	0.7568	0.7544	0.7519	0.9710	0.9263	0.8107	0.5314	0.8845	0.8632	0.8591	0.8595
VARMLP	RRSE	0.5582	0.6579	0.6023	0.6146	0.1922	0.2679	0.4244	0.6841	0.1393	0.1620	0.1557	0.1274
	RAE	0.4510	0.5434	0.4947	0.5474	0.1051	0.1635	0.3102	0.6084	0.0970	0.1171	0.1261	0.0883
	CORR	0.8245	0.7695	0.7929	0.7891	0.9829	0.9655	0.9058	0.7149	0.8708	0.8389	0.8192	0.8679
GRU	RRSE	0.5396	0.5568	0.5618	0.5720	0.2101	0.2818	0.4390	0.4951	0.1150	0.1201	0.1275	0.1380
	RAE	0.3921	0.4142	0.4276	0.4384	0.1168	0.1667	0.2765	0.3043	0.0623	0.0701	0.0789	0.0883
	CORR	0.8461	0.8496	0.8302	0.8241	0.9746	0.9625	0.9119	0.8768	0.8523	0.8361	0.8310	0.8255
TCN	RRSE	0.4852	0.4963	0.5234	0.5287	0.1990	0.2856	0.4138	0.4861	0.1053	0.1123	0.1155	0.1201
	RAE	0.3549	0.3798	0.3838	0.3912	0.1065	0.1544	0.2401	0.2954	0.0589	0.0662	0.0695	0.0752
	CORR	0.8701	0.8578	0.8523	0.8511	0.9788	0.9610	0.9276	0.8721	0.8813	0.8842	0.8752	0.8698
LSTNet	RRSE	0.4777	0.4893	0.4950	0.4973	0.1944	0.2601	0.3408	0.4631	0.0906	0.0974	0.1007	0.1007
	RAE	0.3509	0.3745	0.3749	0.3887	0.0981	0.1491	0.2159	0.2861	0.0519	0.0542	0.0567	0.0549
	CORR	0.8715	0.8627	0.8614	0.8588	0.9823	0.9676	0.9397	0.8794	0.9195	0.9077	0.9045	0.9041
TPA-LSTM	RRSE	0.4487	0.4658	0.4641	0.4765	0.1803	0.2347	0.3234	0.4389	0.0823	0.0916	0.0964	0.1006
	RAE	**0.2901**	0.2999	0.3112	0.3118	0.0918	0.1296	0.1902	0.2727	0.0463	0.0491	0.0541	0.0544
	CORR	0.8812	0.8717	0.8717	0.8629	0.9850	0.9742	0.9487	0.9081	0.9429	0.9337	0.9250	0.9133
MTGNN	RRSE	**0.4162**	0.4435	0.4461	0.4535	**0.1778**	0.2348	0.3109	0.4270	**0.0745**	**0.0862**	**0.0916**	0.0953
	RAE	0.3065	0.3124	0.3218	0.3345	**0.0908**	0.1321	0.1895	0.2787	**0.0446**	**0.0473**	**0.0511**	0.0538
	CORR	**0.8963**	0.8815	0.8794	0.8810	0.9852	0.9726	0.9509	0.9031	**0.9474**	**0.9354**	0.9278	0.9234
HyDCNN (Our model)	RRSE	0.4198	**0.4290**	**0.4352**	**0.4423**	0.1806	**0.2335**	**0.3094**	**0.4225**	0.0832	0.0898	0.0921	**0.0940**
	RAE	0.2969	**0.2989**	**0.3040**	**0.3094**	0.0912	**0.1215**	**0.1786**	**0.2653**	0.0487	0.0479	0.0505	**0.0510**
	CORR	0.8915	**0.8855**	**0.8858**	**0.8819**	**0.9865**	**0.9747**	**0.9515**	**0.9096**	0.9354	0.9329	**0.9285**	**0.9264**

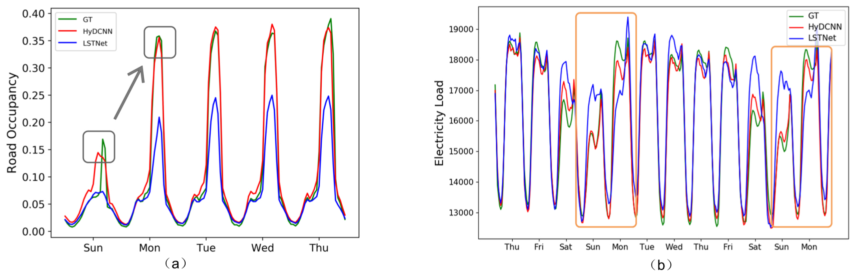


Fig. 6. Visualization of Forecasting results for HyDCNN and LSTNet on different datasets. (a) Traffic dataset with *horizon* = 24. HyDCNN is much more effective capturing the workday peak hours than LSTNet. (b) Electricity dataset *horizon* = 12. HyDCNN learns the daily patterns more effectively than LSTNet.

grid search scheme.[2] For AR, VARMLP we adopt the results provided by [28]. For MTGNN and TPA-LSTM. We adopt their origin results in the papers.

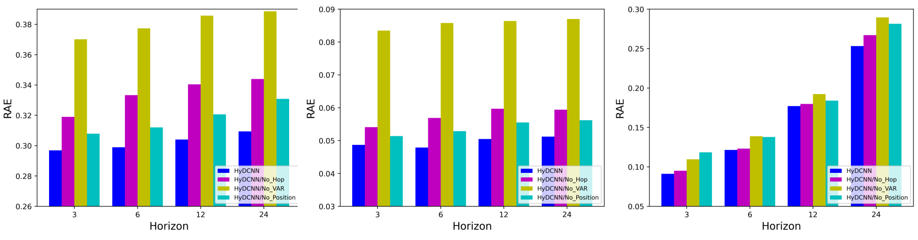
5.3 Experimental Results and Discussions

5.3.1 Results Overview. Table 2 and Table 3 summarize the experimental results of multivariate and univariate datasets in the RRSE, RAE and CORR metrics, respectively. We set *horizon* = $\{3, 6, 12, 24\}$ for all the datasets. Both tables clearly illustrate that our proposed HyDCNN outperforms other competitors in most tasks. Although our HyDCNN does not achieve an overwhelming superiority

[2]https://github.com/laiguokun/LSTNet

Table 3. Experimental results summary of all methods on unitivariate datasets (in RSE, RAE and CORR metrics). Best results are displayed in **boldface**.

dataset		GEFCom(2014)				SML2010			
		horizon				horizon			
methods	metrics	3	6	12	24	3	6	12	24
GRU	RRSE	0.2201	0.3412	0.4102	0.4507	0.1860	0.3181	0.5037	0.6722
	RAE	0.1775	0.2552	0.3486	0.3219	0.1598	0.2581	0.4329	0.6073
	CORR	0.9564	0.9259	0.9126	0.9088	0.9687	0.9432	0.8882	0.7843
LSTNet	RRSE	0.1931	0.2908	0.3609	0.3534	**0.1590**	0.2779	0.4317	0.5908
	RAE	**0.1495**	0.2285	0.3085	0.2906	0.1390	0.2333	0.3757	0.5251
	CORR	0.9812	0.9572	0.9333	0.9364	**0.9897**	0.9661	0.9107	0.8294
HyDCNN-un (Our model)	RRSE	**0.1715**	**0.2323**	**0.2692**	**0.3006**	0.1667	**0.2426**	**0.3451**	**0.5193**
	RAE	0.1533	**0.2035**	**0.2295**	**0.2518**	**0.1384**	**0.2115**	**0.3095**	**0.4965**
	CORR	**0.9866**	**0.9756**	**0.9638**	**0.9542**	0.9877	**0.9705**	**0.9457**	**0.8625**



(a) (b) (c)

Fig. 7. Results of different variants of HyDCNN on different datasets. (a) Traffic dataset. (b) Electricity dataset. (c) Solar-Energy dataset.

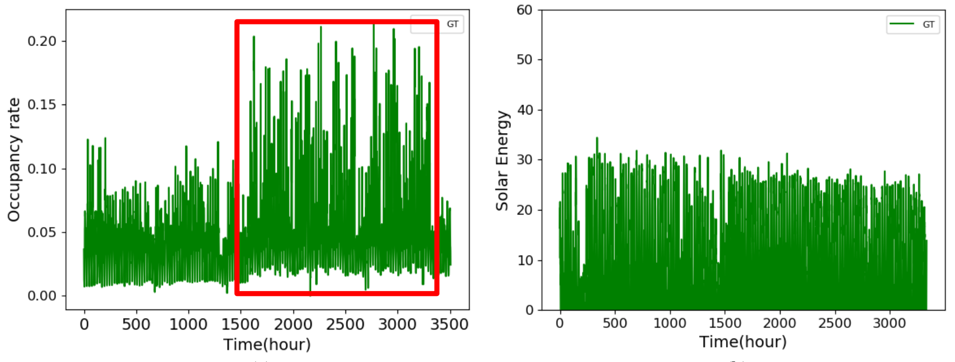
in terms of accuracy comparing with the TPA-LSTM and MTGNN, we still have a great advantage in computational efficiency. TPA-LSTM introduce the attention mechanism, which is quite computation intensive to calculate the variable-wise attention score, with the LSTM. The MTGNN has the time complexity of O(n^2), where n denotes the number of variables, in its graph learning layer. The complexity of our HyDCNN is linear to the number of variables and the computation of CNN is much easier to parallel than RNN.

The visualized results of HyDCNN and LSTNet on the Traffic and the GEFCom datasets are shown in Fig. 6. The results demonstrate that HyDCNN is superior to LSTNet in capturing multiple patterns. For examples, HyDCNN managed to capture the surge from Monday to Sunday on the Traffic dataset as shown in Fig. 6(a) and HyDCNN have learnt the workday and weekend electricity load patterns much more precisely than LSTNet as shown in Fig. 6(b).

Besides, there is one more thing we want to clarify that HyDCNN has demonstrated its superiority on conventional real-world multivariate datasets in Table 2, whereas Table 3 is only a supplement on univariate tasks. Therefore, we compare the challenging methods for univariate datasets only.

5.3.2 Ablation studies. To examine the effectiveness of all the proposing schemes, we conduct a set of ablation studies on the Traffic, Solar Energy, and Electricity datasets. The following notations denote different variants of HyDCNN.

- HyDCNN/No_Hop: HyDCNN variant without the hop hybrid module.
- HyDCNN/No_VAR: HyDCNN variant without VAR components.
- HyDCNN/No_Position: HyDCNN variant without multi-span temporal feature aggregation scheme and position embedding.



(a) (b)

Fig. 8. (a) There is a sudden change in the Traffic dataset from 1600-th hour to 3200-th hour. (b) There is no sudden change in the Solar Energy dataset.

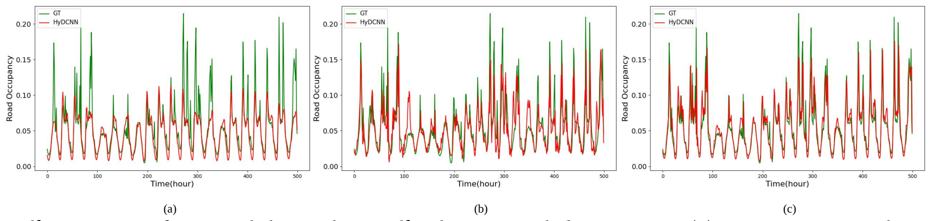


Fig. 9. Effectiveness of AR module on the Traffic dataset with *horizon* = 3. (a) Forecasting results of HyDCNN/No_AR. (b) Forecasting results of AR. (c) Forecasting results of HyDCNN. HyDCNN can capture the sudden changes in time series with the help of the AR module.

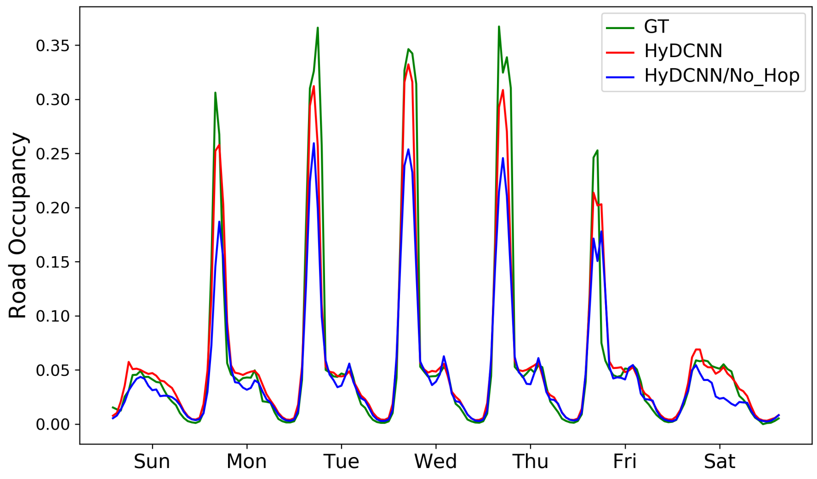


Fig. 10. Effectiveness of the Hop Scheme. Forecasting of HyDCNN (red) and HyDCNN/No_Hop (blue) on the Traffic dataset with *horizon* = 24 vs. the true data (green). HyDCNN can better learn the periodic patterns of daily rush hours with the proposed hop scheme.

The experimental results are presented in Fig. 7. The effectiveness of each proposed scheme in HyDCNN is analyzed as follows.

Effectiveness of Hybrid Mechanism for Linearity and Non-linearity. As shown in Fig. 7, combining with the AR model can improve the forecasting accuracy on all the datasets. It demonstrates the effectiveness of hybrid mechanism. We also can observe that AR components are improves a lot for Traffic and Electricity datasets but improve little on the Solar Energy dataset.

To explain this, we visualize Traffic and Solar Energy datasets in Fig. 8. There are some sudden scale changes in Traffic dataset, but little scale changes in Solar Energy dataset. So we suggest that the AR component may brings the improvement by modeling these scale change with linear combination.

To further study this problem, we extract the records from 2500 to 3000 hours in Fig. 8(a) to represent a time span with scale increase (most spikes in the training set are blow 0.15). We then visualize the output of CNN-only, AR-only, and our hybrid HyDCNN in 9. In Fig. 9(a), the CNN-only model try to fit the ground truth with a regular pattern but failed because of these random changes, mainly because there are very few spikes larger that 0.15 in the training set. In Fig. 9(b), the AR-only model can better capture these random spikes but fail to learn some internal temporal patterns. The hybrid module with AR component nicely makes up for this flaw and achieves significant improvement in Fig. 9(c).

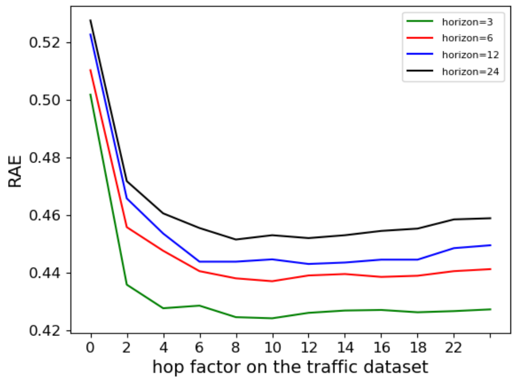


Fig. 11. Performance of different input window size of AR on the traffic dataset. A small window size can bring significant improvement. The best result is achieved when window size is around 10. Moreover, the performance is not sensitive to window size.

To further study how the input window size of AR model influences the performance of HyDCNN. We change the the input window size and evaluate on the traffic dataset. The result is shown in Fig. 11. We notice that even a small window size can bring significant improvement. The best result is achieved when window size is around 10. Moreover, the performance is not sensitive to window size. This result indicates that, the linear dependencies mainly exist within the most recent values

The visualized prediction results of HyDCNN/No_VAR and HyDCNN are shown in Fig. 9, where we can clearly observe that HyDCNN with VAR model shows a great ability for sudden changes.

Effectiveness of Hop Scheme. From Fig. 7, we can observe that adopting the hop hybrid module can improve forecasting performance. We visualize the forecasting of HyDCNN (red) and HyDCNN/No_Hop (blue) in Fig. 10, to demonstrate the ability of HyDCNN to capture the mixtures

Table 4. Running time of calculating a batch of data.

Inference time (ms)	GEFCom	Traffic
GRU	48.37	55.48
LSTNet	8.913	12.51
HyDCNN (Our model)	**4.322**	**5.861**

of sequential and periodic patterns. The visualized prediction results which are presented in Fig.

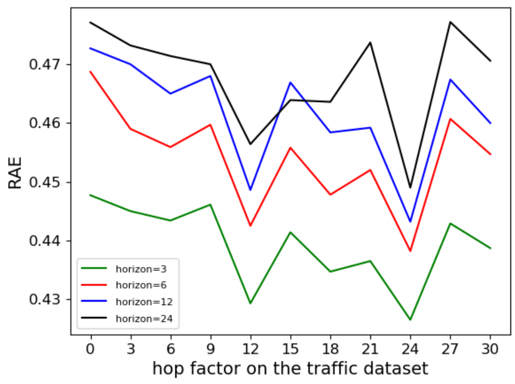


Fig. 12. Performance of different hop factor on the traffic dataset. The best performance is achieved when the hop factor is 24. Traffic dataset is hourly sampled and the result indicates that the improvement is mainly brought by capturing the daily periodic patterns using the hop hybrid module.

10 present that morning and evening rush hours in workdays, free hours in weekend are better captured with the help of hop hybrid module. Obviously, that demonstrates HyDCNN's ability to capture the periodic patterns.

To further study how the hop factor influences the performance of HyDCNN. We change the hop factor and evaluate on the traffic dataset. The result is shown in Fig. 12. The best performance is achieved when the hop factor is 24. Traffic dataset is hourly sampled and the result indicates that the improvement is mainly brought by capturing the daily periodic patterns using the hop hybrid module.

Effectiveness of Capturing Position Information. From Fig. 7, we can observe that position embedding introduces general improvement on all the datasets. The proposed position-aware scheme and the multi-span temporal feature aggregation scheme can be utilized as general strategies on CNN based methods for the problem of time series prediction.

5.4 Inference Time

Table 4 summarizes the inference time of HyDCNN and two other RNN-based methods, GRU and LSTnet, on Traffic and GEFCom datasets. In real-world situations, inference time is more important than training time, and hence, we report the inference time only. For a fair comparison, the parameters are set to the same with the experiments of comparing the prediction accuracy, and the input length for all the methods is set to 336 that means 2 weeks for both datasets. We can observe that HyDCNN is at least 200% faster than the baselines with regard to inference time.

6 CONCLUSIONS

Time series data are ubiquitous and very important in our daily life across various domains. However, how to design accurate and prompt forecasting algorithms is a challenging task, because real-world temporal data often involve both non-linear dynamics and linear dependencies, and always have some mixtures of sequential and periodic patterns. In this work, we propose a novel hybrid framework, termed as HyDCNN, based on position-aware fully Dilated CNN and auto-regressive model, for time series forecasting. HyDCNN is end-to-end trainable and can capture linear dependencies and non-linear dynamics for both the sequential and periodic patterns. Extensive experiments on five real-world datasets have shown that the proposed HyDCNN is better compared with state-of-the-art baselines and is at least 200% than RNN baselines. In the future, we will study how to further capture the spatial dependencies more effectively with CNNs especially on the high-dimensional multivariate time series.

ACKNOWLEDGMENTS

The research was partially funded by National Key Research and Development Program of China (Grant No. 2018YFB0203800), the National Key R&D Program of China (Grant No.2018YFB1003401, 2018YFB0204302), the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant No.61625202), the International (Regional) Cooperation, National Key Research and Development Program of China (2018YFB0204302), NSFC (Grant No. 61772182), Science and Technology on Information Systems Engineering Laboratory and Exchange Program of National Natural Science Foundation of China (Grant No. 61661146006, 61860206011).

REFERENCES

[1] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery* 31, 3 (2017), 606–660.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[3] Mikołaj Bińkowski, Gautier Marti, and Philippe Donnat. 2017. Autoregressive convolutional neural networks for asynchronous time series. *arXiv preprint arXiv:1703.04122* (2017).

[4] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational ....

[5] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. 2018. Dilated convolutional neural networks for time series forecasting. *Journal of Computational Finance, Forthcoming* (2018).

[6] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons.

[7] Li-Juan Cao and Francis Eng Hock Tay. 2003. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks* 14, 6 (2003), 1506–1518.

[8] Cen Chen, Kenli Li, Aijia Ouyang, and Keqin Li. 2018. Flinkcl: An opencl-based in-memory computing architecture on heterogeneous cpu-gpu clusters for big data. *IEEE Trans. Comput.* 67, 12 (2018), 1765–1779.

[9] Cen Chen, Kenli Li, Aijia Ouyang, Zeng Zeng, and Keqin Li. 2018. GFlink: An in-memory computing architecture on heterogeneous CPU-GPU clusters for big data. *IEEE Transactions on Parallel and Distributed Systems* 29, 6 (2018), 1275–1288.

[10] Cen Chen, Kenli Li, Sin G Teo, Xiaofeng Zou, Kang Wang, Jie Wang, and Zeng Zeng. 2019. Gated Residual Recurrent Graph Neural Networks for Traffic Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 485–492.

[11] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1251–1258.

[12] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[13] Marco Cuturi. 2011. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 929–936.

[14] Sakyasingha Dasgupta and Takayuki Osogami. 2017. Nonlinear dynamic Boltzmann machines for time-series prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[15] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 933–941.

[16] Jan G De Gooijer and Kuldeep Kumar. 1992. Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting* 8, 2 (1992), 135–156.

[17] Robert F Engle. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society* (1982), 987–1007.

[18] Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A Convolutional Encoder Model for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 123–135.

[19] Clive William John Granger and Allan Paul Andersen. 1978. *An introduction to bilinear time series models*. Vandenhoeck & Ruprecht.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[22] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J Hyndman. 2016. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond.

[23] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[24] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. 2019. DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2129–2132.

[25] Ashu Jain and Avadhnam Madhav Kumar. 2007. Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing* 7, 2 (2007), 585–592.

[26] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[27] Kyoung-jae Kim. 2003. Financial time series forecasting using support vector machines. *Neurocomputing* 55, 1-2 (2003), 307–319.

[28] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 95–104.

[29] Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S Dhillon. 2019. Similarity Preserving Representation Learning for Time Series Clustering.. In *IJCAI*, Vol. 19. 2845–2851.

[30] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[31] Tao Lin, Tian Guo, and Karl Aberer. 2017. Hybrid neural networks for learning the trend in time series. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2273–2279.

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.

[34] Helmut Lütkepohl. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.

[35] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2014. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2014), 865–873.

[36] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437* (2019).

[37] Syama Sundar Rangapuram, Matthias Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. In *Proceedings of the 32nd international conference on neural information processing systems*. 7796–7805.

[38] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.

[39] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[40] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning* 108, 8-9 (2019), 1421–1441.

[41] Osamah Basheer Shukur and Muhammad Hisyam Lee. 2015. Daily wind speed forecasting through hybrid KF-ANN model based on ARIMA. *Renewable Energy* 76 (2015), 637–647.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. (2017).

[43] Paul J Werbos et al. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[44] Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. 2018. Random warping series: A random features method for time-series embedding. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 793–802.

[45] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. *arXiv preprint arXiv:2005.11650* (2020).

[46] Subin Yi, Janghoon Ju, Man-Ki Yoon, and Jaesik Choi. 2017. Grouped convolutional neural networks for multivariate time series. *arXiv preprint arXiv:1703.09938* (2017).

[47] Fransisco Zamora-Martinez, Pablo Romeu, Pablo Botella-Rocamora, and Juan Pardo. 2014. On-line learning of indoor temperature forecasting models towards energy efficiency. *Energy and Buildings* 83 (2014), 162–172.

[48] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. 1998. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting* 14, 1 (1998), 35–62.

[49] G Peter Zhang. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50 (2003), 159–175.