Information Processing and Management 59 (2022) 102987

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Improved sales time series predictions using deep neural networks with spatiotemporal dynamic pattern acquisition mechanism

Daifeng Li [a,*], Kaixin Lin [a], Xuting Li [a], Jianbin Liao [a], Ruo Du [b], Dingquan Chen [a], Andrew Madden [c]

[a] *School of Information Management, Sun Yat-Sen University, Guangzhou, China*
[b] *Galanz Inc., Shenzhen, China*
[c] *Information School, University of Sheffield, Sheffield, UK*

ARTICLE INFO

Keywords:
Sales prediction
Spatiotemporal dynamic kernel
Simultaneous regression
Hierarchical attention
Dynamic detection and alignment

ABSTRACT

The ability to predict product sales is invaluable for improving many of the routine decisions essential for the running of an enterprise. One significant challenge of sales prediction is that it is hard to dynamically capture changing dependent patterns along the sales time line, because sales are often influenced by complicated and changeable market environment. To address this issue, we model sales prediction as a task of multivariate time series (MTS) prediction, and propose a Spatiotemporal Dynamic Pattern Acquisition Mechanism (SDPA), which comprises four components, described below: (1) In the processing of input data: A Spatiotemporal Dynamic Kernel (SDK) component is designed for MTS to effectively capture different dependent correlation patterns during different time periods. (2) In terms of model design: A Simultaneous Regression (SR) component is proposed to dynamically detect stable correlations by using co-integration based dynamic programming over different time periods. (3) A novel Hierarchical Attention (HA) component is designed to incorporate SDK to detect spatiotemporal attention patterns from the captured dynamic correlations. (4) In the design of loss function, A Change Sensitive and Alignment component (DC) is proposed to provide more future information based on future trend correlations for better model training. The four components are incorporated into a unified framework by considering Homovariance Uncertainty (HU). This is referred to as SDPANet and contributes to model training and sales prediction. Extensive experiments were conducted on two real-world datasets: Galanz and Cainiao, and experimental results show that the proposed method achieves statistically significant improvements compared to the most state-of-the-art baselines, with average 41.5% reduction on RMAE, average 39.5% reduction on RRSE and average 46% improvement on CORR. Experiments are also conducted on two new datasets, which are Traffic and Exchange-Rate from other fields, to further verify the effectiveness of the proposed model. Case studies show that the model is capable of capturing dynamic changing patterns and of predicting future sales trends with greater accuracy.

1. Introduction

The prediction of product sales is an important task for enterprises. Consequently, it has been widely studied in both academia and industry (Ekambaram et al., 2020; Shi et al., 2021). Accurate forecasts of sales can improve revenues by, for example, streamlining

[*] Corresponding author.
E-mail addresses: lidaifeng@mail.sysu.edu.cn (D. Li), linkx5@mail2.sysu.edu.cn (K. Lin), ruodu56@hotmail.com (R. Du), chendq@mail.sysu.edu.cn (D. Chen).

https://doi.org/10.1016/j.ipm.2022.102987
Received 8 January 2022; Received in revised form 20 May 2022; Accepted 22 May 2022
Available online 14 June 2022

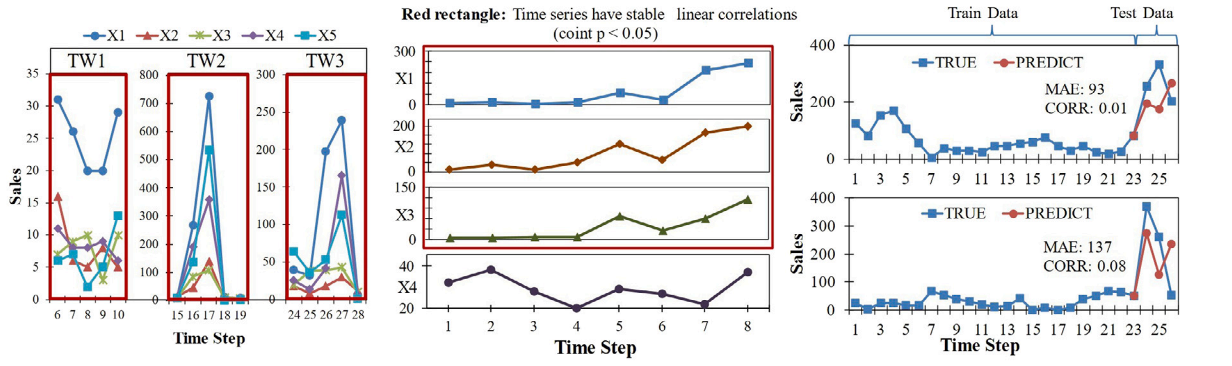
inventory management. A key challenge of product sales forecasting is to develop models capable of effectively capturing the complex, non-linear dependencies, not only between time steps but also between a variety of variables (Das & Ghosh, 2017; Shi et al., 2020a; Tang et al., 2020). This could be regarded as a multivariate time series forecasting problem. Examples of relevant dependencies include the arrival of similar products, new promotion strategies from competitors, the impact of the timing of promotions, etc. Traditional time series foresting methods, such as vector auto-regression (VAR) (Box, Jenkins, Reinsel, & Ljung, 2015) and Gaussian process (GP) (Roberts et al., 2019), usually assume certain distributions, and ignore dependencies between variables (Nguyen et al., 2020).

In recent years, using Deep Neural Network (DNN) for multivariate time series (MTS) predictions has been successfully used to predict financial trends, analyze traffic jams, and forecast electricity consumption. DNN have advantages arising from their flexibility in capturing non-linearity. One widely-used approach is to use a Recurrent Neural Network (RNN) (Dasgupta & Osogami, 2017; Lai, Chang, Yang, & Liu, 2018). Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) combining attention mechanisms (Vaswani et al., 2017) can further improve the capability of RNNs to help model temporal patterns between fragments of input time series (Fan et al., 2019; Gao, Kong, Lu, Bai, & Yang, 2018; Huang, Wang, Wu, & Tang, 2019). Modeling a mixture of influences of correlation patterns (Gao et al., 2018; Qin et al., 2017; Ye, Luo, Xiao, & Ma, 2020), for which traditional approaches may fail, is an important research direction for MTS forecasting. The approach uses a Convolutional Neural Network (CNN) to capture local correlations between different time series. It then applies LSTM/GRU to model long and short-term temporal patterns (Lai et al., 2018). Other technologies that have performed well include transformer (Lim, Ark, Loeff, & Pfister, 2021; Zhou et al., 2021a), tensor decomposition (Shi et al., 2020a), adversarial training (Tang et al., 2020), extreme value prediction (Laptev, Yosinski, Li, & Smyl, 2017), and explainable time series prediction (Oreshkin, Carpov, Chapados, & Bengio, 2019). The deep learning model not only takes into account the advanced nature of the model but also combines some practical application scenarios. For example, BHT-ARIMA (Shi et al., 2020b) is designed based on deep learning and tensor decomposition for predicting the sales of mobile phones and laptops.

However, existing deep neural network-based research seldom considers the changing dynamic correlations along the timeline, especially in the area of sales predictions. Fig. 1(a) indicates that there exists complex non-linear correlations between sales time series. A sales MTS contains 5 variables (X1~X5). Each red rectangle (TW1, TW2 and TW3) represents a time window. In TW1, the sales of X1~X5 are very close, while in TW2 and TW3, changes in the external environment cause sales of same variables (X1, X3 in TW2; X1, X3 and X4 in TW3) to increase significantly. The weight-sharing strategy used by traditional CNN does not properly reflect the dynamic external changes, because it uses the same kernel to extract dependent correlations. A practical solution is to use dynamic weights to reflect different external environments. Dynamic convolution (Chen et al., 2020; Hu, Shen et al., 2020) is a feasible solution. However, current research of dynamic kernel is mainly applied in Computer Vision (CV) domain and focuses on detecting correlations between different kernels or channels to extract rich semantic information from images. Different from the existing researches, we mainly focus on investigating how to capture dynamic correlations for each element of MTS from a more microscopic perspective, because an element or "cell" in MTS represents specific product sales at a certain time point and is essential important for MTS prediction. The idea of dynamic kernel could be considered as a more dynamic and flexible parameter assignment in a larger parameter space. Many researches in deep learning have shown a larger parameter space strengthens generalization capability, because the model has more parameter selection space in which to fit the complex nonlinear patterns (Cen et al., 2019; Liang, Li, & Madden, 2020).

Fig. 1(b) shows that there exists dynamic linear correlations between sales time series. For the sales MTS with 4 time series $X_1 \sim X_4$, the red rectangle shows that X_1, X_2 and X_3 have stable linear correlations, which may not be detected using existing deep neural network-based methods; and the learned information at time step 7 from X_4 may adversely affect future predictions of X_1, X_2 and X_3. We have observed that there are a lot of dynamic linear relationships hidden in the sales MTS, because product sales usually follow some general market rules in the case of less external interference. However, these linear correlations change dynamically over time, so how to effectively detect the linear correlations from MTS is still a challenging research subject.

Fig. 1(c) shows an example of sales prediction using MLCNN (Cheng, Huang, & Zheng, 2020) by adopting L1-loss. Researches such as MLCNN consider that, modeling the dependent correlations between the values of future times (ex. $t+1$, $t+2$, …) will provide a significant positive contribution to time series predictions. Though the accumulated MAE of the two selected sales time series for predicting sales from time 25 to 26 are not big (Accumulated MAE is 93 in upper sub-graph and is 137 in lower sub-graph), CORR is relatively small. The reason is that L1-loss only focuses on the overall deviations between the predicted and the true values based on their corresponding future time points, and cannot accurately capture the future trends of target time series. As introduced in related studies (Cheng et al., 2020; Li, Xu, Taylor, Studer, & Tom, 2018; Yu, Jiang, Wang, Cao, & Huang, 2016), if the loss function cannot consider the various causes of loss more comprehensively, the model may not make proper use of the captured patterns. For example, for the task of object identification in CV domain, the model should locate the four vertices of the object area. The predicted area will deviate from the actual area if we do not consider the correlations between these vertices. Thus an IoU loss is proposed to more accurately align the predicted and true areas (Yu et al., 2016). Similarly, in the task of MTS prediction, existing researches usually adopt L1 or L2-loss. Therefore, the methods cannot make proper use of future dependent correlations to predict future trends (ex. Sales predictions are very accurate at some times, but there are large deviations at other times). As seen in Fig. 1(c): according to Dynamic Time Warping (DTW) (Sakoe & ChibaLim, 1990), the values at time period 23~25 of true sales and at time period 23~26 of predicted sales have similar patterns of upward trends. Traditional loss functions may fail to capture these patterns because the time length of the two future time series (23~25 vs 23~26) is inconsistent, and there is a position deviation between the peaks (The peak of the true sales is at 25, the peak of the predicted sales is at time 26). In summary, due to the existence of deformation, the idea of IoU loss cannot be directly used in sales prediction. The main challenge is that we do not know which part



(a) Example: Dynamic changing patterns (b) Example: Dynamic stable correlations (c) Example: Trend prediction errors by using L1-loss.

Fig. 1. Examples of main challenging research subjects in sales predictions.

of the predicted and true sales should be aligned. Thus, we need to design a new loss function to find parts with similar patterns, and make better use of future dependent correlations for model training and sales predictions.

The above three challenges are particularly significant in sales time series prediction, which can be considered as specific characteristics of sales time series. Because sales time series are more susceptible to external uncertainties, which will change the inner-correlations between sales time series dynamically. External uncertainties are caused by influential factors that cannot be determined or observed in advance. External uncertainties would include temporarily decided regional promotion activities, the listing of new competitive products, etc. Earlier research considered that MTS could use patterns of commonness, or difference between time series, to reflect the impact of external changes on the predicted value (Cheng et al., 2020; Lai et al., 2018; Qin et al., 2017). However, a key premise when making a judgment on the future is that we can obtain sufficient meaningful linear and non-linear patterns from historical data. This is easier to achieve with some time series than others. For example, Traffic data has easily identifiable periodic characteristics; Exchange-Rate time series are strongly correlated with each other, with the correlations being less affected by external uncertainties. By contrast, the dynamic changing correlations of sales time series make them particularly difficult to model. In this research, a novel Spatiotemporal Dynamic Pattern Acquisition (SDPANet) model is proposed to address the challenges mentioned above. The main innovations of the model are summarized as below:

- The proposed Spatiotemporal Dynamic Pattern Acquisition (SDPA) mechanism incorporates four novel components into a unified framework. This helps to capture changing correlation patterns in multivariate sales time series, and improves the accuracy of sales predictions.
- A novel Spatiotemporal Dynamic Kernel (SDK) component is proposed for better capturing dynamic correlation patterns. Unlike previous studies are mainly applied in Computer Vision (CV) domain, this designs a spatiotemporal weight matrix for each element of time point in MTS. The output of SDK can be fed into a Hierarchical Attention (HA) component to detect spatiotemporal attention patterns.
- Simultaneous Regression (SR) component uses a co-integration test based dynamic programming algorithm to detect local stable linear correlations in a time window scanned along the timeline. This helps to overcome one of the limitations of existing Deep Neural Network, that the scale of outputs is not linear sensitive to the scale of inputs. Different from existing auto regression (AR) solution, SR mainly focuses on detecting plenty of dynamic linear correlations between time series from MTS.
- A Dynamic Change Detection and Alignment (DC) component is proposed to provide more future information based on future trend correlations for the model training to capture useful correlation patterns, and it can further adjust the prediction value for more accurate future trend predictions based on the captured correlation patterns.

Codes and experimental datasets are available at: https://github.com/inksyy/SDPANet

2. Related work

2.1. Product sales prediction

The accurate prediction of product sales is clearly of value to enterprises. It can, for example, help decision makers to optimize inventory management. Resulting in considerable savings, meanwhile, realize "competition for customers' time" (Shen, Yuan, Wu, & Pei, 2018). Traditional sales time series predictions are based on auto-regressive integrated moving averages (ARIMA), which detect correlations in linear stationary time series (Gao et al., 2018; Lai et al., 2018). For example, Box et al. (2015) analyzed the results of tests for unit root non-stationarity in ARIMA processes. They modeled the state space representation of ARMA models and discussed its use for likelihood estimation and forecasting. Methods based on traditional machine learning rely mainly on ARIMA, Vector Auto Regression, Support Vector Machines (SVM) (Mellit, Pavan, & Benghanem, 2013), Factorization Machines (FM) (Chen, Liu, Zheng, & Yu, 2018), etc. Shen et al. (2018) incorporated popular machine learning models such as ARIMA, Xgboost, SVM, Gaussian Process etc, into a unified ensemble model to forecast product sales for JD.com, one of the largest e-commerce companies in China. Bello-Orgaz et al. (2020) used classical unsupervised machine learning techniques, such as temporal clustering and hidden Markov models, to extract collective temporal behavior patterns of the dynamics of customers over time.

In recent years, deep neural network-based methods have been used to improve sales predictions based on time series predictions, Fan et al. (2019) proposed a novel data-driven model to explicitly learn constructing hidden patterns' representations with deep neural networks and attending to different parts of the history for forecasting the future. Shi et al. (2020a) incorporated multi-way delay embedding transform (MDT) tensors and tensor ARIMA into a unified framework to capture the intrinsic correlations among multiple time series. They used their model to predict sales of HUAWEI computers. Ekambaram et al. (2020) applied attention based multi-modal time-series forecasting to new products in a dataset from a famous fashion house. All the multiple time-series were modeled together based on product images and optional attributes. Kaya, Yılmaz, Yaslan, Öğüdücü, and Çıngı (2021) adopted Attention-LSTM to realize tourism demand forecasting, the use of NN embedding and K-means algorithms for feature processing can significantly improve model performance.

2.2. *Deep Neural Network based MTS predictions*

Deep Neural Networks (DNNs) have attracted increasing attention in the field of multivariate time series (MTS) forecasting, including prediction of financial trends, analysis of traffic flows, and forecasts of electricity consumption. One popular learning framework using DNN based methods is CNN-LSTM/GRU. LSTNET, for example, models long–short term patterns based on a CNN-GRU framework. To do so, it uses a recurrent-skip component to model long-term temporal patterns which can be controlled by assigning parameter p (Lai et al., 2018). Dual-Stage Attention-Based Recurrent Neural Nets (DA-RNN) (Qin et al., 2017) can also model long-term temporal patterns based on Multivariate Time Series. However, they are thought not to give sufficient consideration to spatial correlations among different components of exogenous data. Dual Self-Attention Network (DSANET) (Huang et al., 2019) adopts CNN to model global and local temporal patterns, and uses a self-attention mechanism to extract sequence features. MTNet designs a memory component, and incorporates it with three separate encoders and an auto-regressive component; it then trains all the components jointly. MLCNN (Cheng et al., 2020) provides a near and distant fusion vision method, to improve predictive performance by adopting a multi-task learning framework and fusing forecasting information. Spectral time Graph Neural Network (StemGNN) (Cao et al., 2020) combines Graph Fourier Transform (GFT) and Discrete Fourier Transform (DFT) into an end-to-end framework. After passing the input data through GFT and DFT, the spectral representations hold clear patterns and can be predicted effectively by convolution and sequential learning modules.

The research of Deep Neural Network based time series prediction is also applied in many areas. For example, Christopher Westland, Mou, and Yin (2019) designed a deep learning model with tuned hyper-parameters to better understand customers' unique habitual behaviors and predict their behaviors in bicycle sharing. Li, Wu, and Wang (2020) adopted LSTM model to incorporate both stock indicators and news sentiments for stock market trend prediction based on a finance domain specific sentiment dictionary. Hu, Yang et al. (2020) proposed a novel deep learning model to understand and predict electricity-theft Behavior via multi-source Data. Song, Lin, Guo, and Wan (2020) proposed a Spatialtemporal Synchronous Graph Convolutional Networks (STSGCN) for network data forecasting. Liang, Mao, Lu, Bai, and Gang (2021) adopted deep neural network models combining with bibliometric indicator to realize emerging research topic prediction. Yakhchi et al. (2022) proposed a novel Convolutional Attention Network for sequential recommendations.

Although these advanced studies have succeeded in enhancing MTS prediction, they seldom consider capturing dynamical changing patterns in a more effective way, and model sales time series at a more microscopic level. Besides, few address the issue how to effectively make use of the detected dynamic patterns for model learning. In this research, a novel SDPANet model is proposed to address the two issues mentioned above.

3. Model descriptions

3.1. *Problem statement*

Assume there are N products $X = \{X^1, \ldots, X^i, \ldots, X^N\}$ of different models in a warehouse. The sales time series of ith product through time span $1 \sim t$ is $X^i = \{X^i_1, \ldots, X^i_t\}$. Each X^i has M feature time series $F^i = \{f^{i,1}, \ldots, f^{i,j}, \ldots, f^{i,M}\}$. Features include users' visiting records of the product, discount rate, promotion activity, etc. Each X^i and its feature set F^i could be encoded into a new sales time series $X^i_{1\sim t} = SDK(X^i, F^i)$, where SDK (Spatiotemporal Dynamic Kernel) will be introduced in Section 3.2.1. Therefore, the main aim of this research is to predict the sales Y_{t+1} of all N products at time $t+1$, where $Y_{t+1} = \{Y^1_{t+1}, \ldots, Y^N_{t+1}\}$. The objective function can be described as $\widetilde{Y}_{t+1} = SDPA(X^{1\sim N}_{1\sim t})$, and $X^{1\sim N}_{1\sim t}$ can be seen as a matrix to representing a multivariate time series, where rows represent spatial information (the new sales embedding of N products at time t), and columns represent temporal information (the new encoded sales time series with their features from time 1 to t). Assume the real sales at time $t+k$ is Y_{t+k}, then the target of the prediction task is to minimize the deviation between Y_{t+k} and $\widetilde{Y}_{t+k}$.

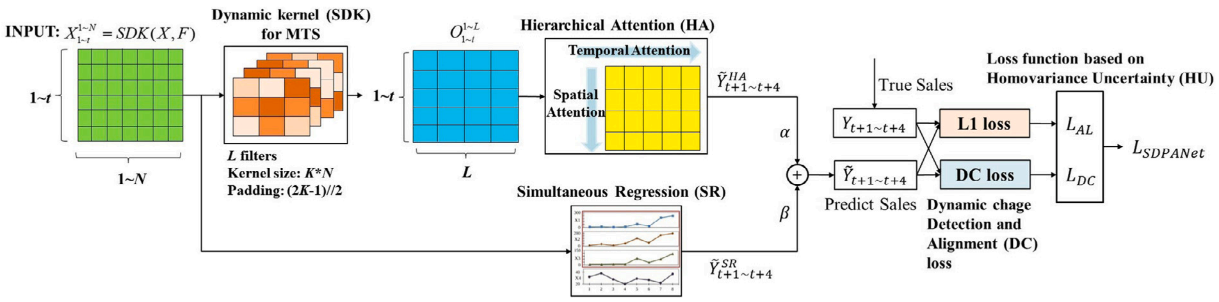

INPUT: $X_{1\sim t}^{1\sim N} = SDK(X, F)$
Dynamic kernel (SDK) for MTS
$O_{1\sim t}^{1\sim L}$
Hierarchical Attention (HA)
Temporal Attention
Spatial Attention
$\widetilde{Y}_{t+1\sim t+4}^{HA}$
True Sales
Loss function based on Homovariance Uncertainty (HU)
$Y_{t+1\sim t+4}$
L1 loss
L_{AL}
$L_{SDPANet}$
$\widetilde{Y}_{t+1\sim t+4}$
DC loss
L_{DC}
Predict Sales
Dynamic chage Detection and Alignment (DC) loss
α
β
1~t
1~N
L filters
Kernel size: K*N
Padding: (2K-1)//2
L
Simultaneous Regression (SR)
$\widetilde{Y}_{t+1\sim t+4}^{SR}$

Fig. 2. The Framework of SDPANet.

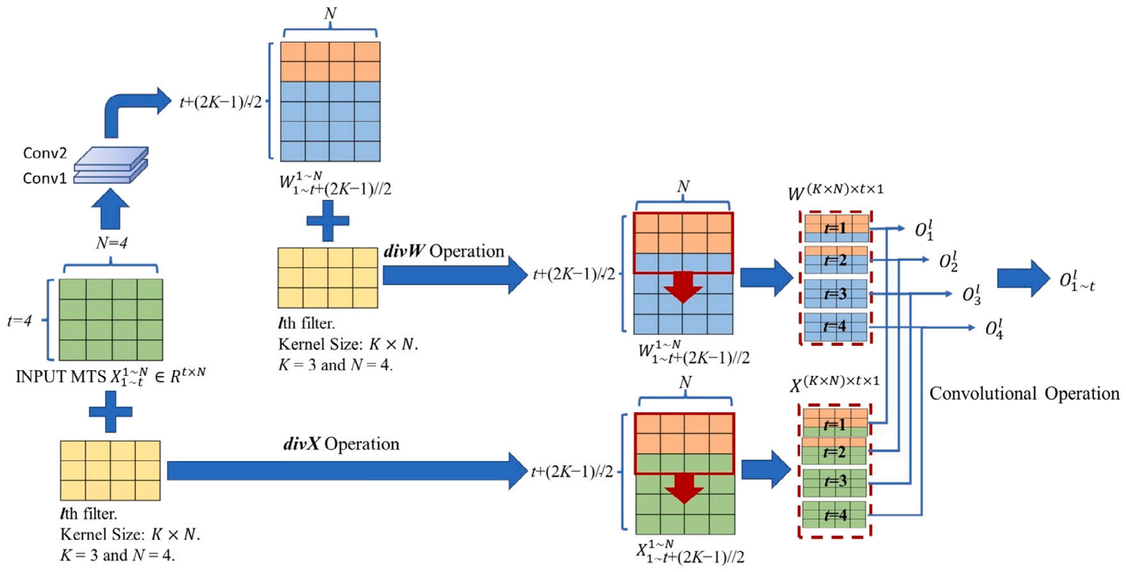

N
t+(2K−1)//2
Conv2
Conv1
$W_{1\sim t+(2K-1)//2}^{1\sim N}$
N=4
t=4
INPUT MTS $X_{1\sim t}^{1\sim N} \in R^{t\times N}$
divW Operation
lth filter.
Kernel Size: $K \times N$.
K = 3 and N = 4.
$W^{(K\times N)\times t\times 1}$
t=1
t=2
t=3
t=4
O_1^l
O_2^l
O_3^l
O_4^l
$O_{1\sim t}^l$
$X^{(K\times N)\times t\times 1}$
Convolutional Operation
divX Operation
$X_{1\sim t+(2K-1)//2}^{1\sim N}$

Fig. 3. Spatiotemporal Dynamic Kernel (SDK).

3.2. Model framework

In Fig. 2, SDPANet incorporates 4 components: Spatiotemporal Dynamic Kernel (SDK), Simultaneous Regression (SR), Hierarchical Attention component (HA), and Dynamic Change Detection and Alignment (DC) component, into a unified framework for sales predictions. The combination of SDK and HA is used to capture dynamic non-linear correlations from MTS. The SR component is used to capture dynamic linear correlations. The DC component can supplement more valuable information to the training of SDK, HA and SR from the perspective of future trend correlations. The predicted sales $\widetilde{Y}_{t+1}^{HA}$ is calculated by SDK and HA, and $\widetilde{Y}_{t+1}^{SR}$ is calculated by SR. The $\widetilde{Y}_{t+1}^{HA}$ and $\widetilde{Y}_{t+1}^{SR}$ will be integrated by L1-loss and DC loss to generate two loss functions, which are L_{AL} and L_{DC} respectively. Finally, L_{AL} and L_{DC} are combined by considering Homovariance Uncertainty (HU) to generate the final loss function $L_{SDPANet}$. The following sections will introduce each component in detail.

3.2.1. Spatiotemporal Dynamic Kernel (SDK) component

Earlier studies overlook the unique characteristics of sales time series, as sales time series is a dynamic process, and will be influenced by complicated environment apart from the timeline, so different local spatiotemporal areas may contain different dependent correlations. Different from existing researches related with dynamic kernel, which are mainly applied in CV domain, a novel SDK is designed for MTS predictions and focuses on assigning specific weight to each element or "cell" of input X. Fig. 3 provides an illustrative representation of the SDK component.

For the input multivariate time series (MTS) $X_{1\sim t}^{1\sim N}$, assume L CNN filters are assigned: for the lth filter, kernel size is $K \times N$. The main idea of SDK is to design a dynamic weight matrix, and to use this to generate a set of $K \times N$ matrices, each of which represents the weight changes of the kernel in the specific area of $X_{1\sim t}^{1\sim N}$. The weights change as the kernel moves in $X_{1\sim t}^{1\sim N}$ reflecting changes in the external environment. In this study, the dynamic weight matrix is based on a two-layer convolutional network, the two layers of which are named as conv1 and conv2 with kernel size as 1×1. The 1×1 kernel provides a dynamic weight for each "element" X_i^j in matrix $X_{1\sim t}^{1\sim N}$, and enhance the non-linear representations of each "element". The design of the weight matrix is shown in formula (1):

$$W_{1\sim t+(2K-1)//2}^{1\sim N} = conv2(conv1(X_{1\sim t}^{1\sim N}, ks = 1), ks = 1, ph = K//2, str = 1) \tag{1}$$

where $W_{1\sim t+(2K-1)//2}^{1\sim N}$ is a dynamic weight matrix with $1 \sim N$ columns and $1 \sim t + (2K-1)//2$ rows. $ks = 1$ represents the kernel size is 1×1, and $//$ represents an integer division operation. The weight matrix extends both columns and rows of $X_{1\sim t}^{1\sim N}$ by $(2K-1)//2$ based on the $ks = kernel\_size$, $ph = padding\_height$ and str = stride. The purpose of assigning $(2K-1)//2$ for rows is to ensure that the height of SDK convolutional output is fixed as t. The convolutional layers conv1 and conv2 generate the dynamic weight matrix, and assigns dynamic non-linear weight for each "element". Obviously, the dynamic weight is highly correlated with the value of each "element". Next, we construct the convolution operation between $W_{1\sim t+(2K-1)//2}^{1\sim N}$ and the input $X_{1\sim t}^{1\sim N}$. The two matrix are separately divided into t $K \times N$ matrices $W^{(K\times N)\times t}$ and $X^{(K\times N)\times t}$ by assigning kernel size as $K \times N$. The expression is shown below:

$$W^l = W^{(K\times N)\times t} = divW(W_{1\sim t+(2K-1)//2}^{1\sim N}, K \times N, 0, 1) \tag{2}$$

$$X^l = X^{(K\times N)\times t} = divX(X_{1\sim t}^{1\sim N}, K \times N, (2K-1)//2, 1) \tag{3}$$

where l represents the lth filter, function $divW$ is used to divide $W_{1\sim t+(2K-1)//2}^{1\sim N}$ into t $K \times N$ matrices $W^l = W^{(K\times N)\times t}$ by assigning kernel size as $K\times N$, padding as 0 and stride as 1. Function $divX$ divides $X_{1\sim t}^{1\sim N}$ into t $K\times N$ matrices $X^l = X^{(K\times N)\times t}$ by assigning kernel size as $K\times N$, padding as $(2K-1)//2$ and stride as 1. The ith $K\times N$ matrix $W^l(i)$ of $W^{(K\times N)\times t}$ can be seen as the weight distribution of the dynamic kernel when it moves to the ith $K \times N$ region $X^l(i)$ of input $X_{1\sim t}^{1\sim N}$, where $i \leq t$. This is the main difference from traditional CNN with weight-sharing strategy, because the weights of the dynamic kernel in all t $K \times N$ areas are totally different. In order to use the t kernel weights to realize convolution operation on $X_{1\sim t}^{1\sim N}$, $X_{1\sim t}^{1\sim N}$ is first extended to $X_{1\sim t+(2K-1)//2}^{1\sim N}$ by using padding operations to align with $W_{1\sim t+(2K-1)//2}^{1\sim N}$, and then divided into t $K \times N$ sub-matrices $X^l = X^{(K\times N)\times t}$ by using $divX$. Thus, the ith convolution operation of SDK is the element-wise product between the ith kernel $W^l(i)$ and ith sub-matrices $X^l(i)$, where $(i \leq t)$. Formula (4) shows the SDK convolution operation on $X_{1\sim t}^{1\sim N}$ with kernel size as $K \times N$.

$$O_{1\sim t}^l = [W^l(i) \odot X^l(i)] = [W^{(K\times N)\times t}(:,:,i) \odot X^{(K\times N)\times t}(:,:,i)] \; for \; i \in \; t \tag{4}$$

where $O_{1\sim t}^l \in R^{t\times 1}$ is the convolution output of lth filter with kernel size as $K \times N$; $\odot$ is element-wise product; $(:,:,i)$ represents the ith $K \times N$ kernel or sub-matrices. The output of each element-wise product is appended to list $[\ldots]$ to generate the one-dimension vector $O_{1\sim t}^l$ with length t.

The output of the lth filter is $O_{1\sim t}^l$. For all the L filters, $O_{1\sim t}^L \in R^{t\times L}$ is the stack of all $O_{1\sim t}^l$ used to form a new matrix with row t and column L. The expression of the output of all L filters is shown in formula (5):

$$O_{1\sim t}^L = Stack_{l=0}^L(O_{1\sim t}^l) \tag{5}$$

Formula (1)~(5) could be defined as a mapping function: $O_{1\sim t}^L = F(X_{1\sim t}^{1\sim N})$. We adopt 4-layer mapping function F to model future-vision based local correlations from time $t+1$ to $t+4$, which could be seen in formula (6):

$$\begin{aligned}
(O_{1\sim t}^L)_{t+1} &= F_0(X_{1\sim t}^{1\sim N}) \\
(O_{1\sim t}^L)_{t+2} &= F_1((O_{1\sim t}^L)_{t+1}) \\
(O_{1\sim t}^L)_{t+3} &= F_2((O_{1\sim t}^L)_{t+2}) \\
(O_{1\sim t}^L)_{t+4} &= F_3((O_{1\sim t}^L)_{t+3})
\end{aligned} \tag{6}$$

where $(O_{1\sim t}^L)_{t+1}$, $(O_{1\sim t}^L)_{t+2}$, $(O_{1\sim t}^L)_{t+3}$ and $(O_{1\sim t}^L)_{t+4}$ are construal of local dependent correlations of all time series variables at time $t+1$, $t+2$, $t+3$ and $t+4$. F_0, F_1, F_2 and F_3 are 4-layer mapping functions based on SDK component.

Compared with Dynamic Convolution (Chen et al., 2020; Hu, Shen et al., 2020). The following content highlights the differences between SDK and Dynamic Convolution. For a MTS, one of its $K \times N$ matrix is $X_{i\sim i+K}^{1\sim N}$ $(i \leq t-K)$. Dynamic convolution (Chen et al., 2020) assigns L filters with kernel size $K\times N$, in which the lth kernel is $W^l(X_{1\sim t}^{1\sim N})$ weight-sharing matrix, and the attention weight is π_l. Then the convolution operation of $X_{i\sim i+K}^{1\sim N}$ can be expressed as:

$$O_i = \sum_{l\in L} \pi_l \times (W^l(X_{1\sim t}^{1\sim N}) \cdot X_{i\sim i+K}^{1\sim N}) \tag{7}$$

SDK assigns specific weight for each time point, and makes convolution operations based on aggregation of all time points in the $K \times N$ matrix. Assume the weight-specific matrix is $W^l(X_{i\sim i+K}^{1\sim N})$ (which could be derived from formula (2)), then SDK convolution operation can be expressed as:

$$O_i^l = W^l(X_{i\sim i+K}^{1\sim N}) \cdot X_{i\sim i+K}^{1\sim N} \tag{8}$$

By comparing formulas (7) and (8), we can see that SDK focuses on more a microscopic level because it directly calculates specific weight W^l for each element in $X_{i\sim i+K}^{1\sim N}$. Each filter l obtains a specific value O_i^l at time i. Dynamic convolution focuses on relationships between global $W^l(X_{1\sim t}^{1\sim N})$ weight-sharing matrix and local $X_{i\sim i+K}^{1\sim N}$ matrix, thus its output, O_i is a single value, which is

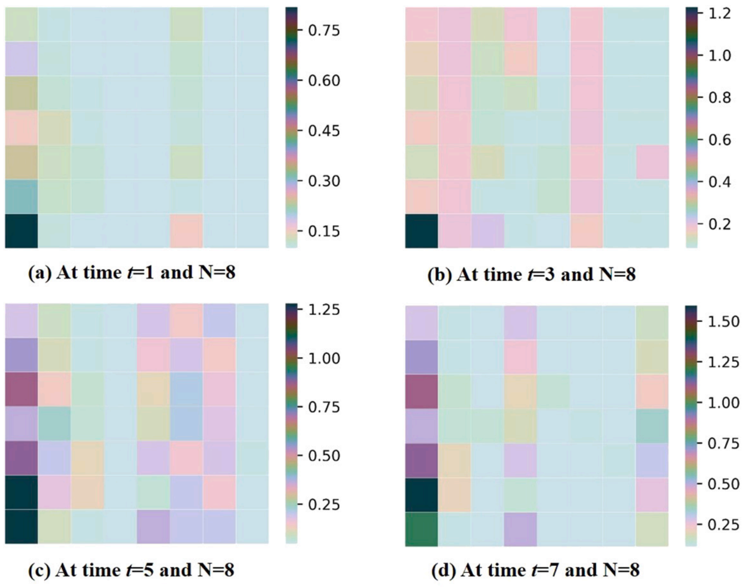


Fig. 4. Visualization of weight changes of SDK kernel with time t. The Input MTS contains 8 time series. The SDK kernel size is set at 8 × 8. We can observe that the kernel weights are dynamically changed when the time is changed. When batch size is 64, the SDPANet needs 1.69 s to complete one iteration, and SDK only needs 0.00149 s.

the weighted summation of all L convolution operations with weight-sharing. This method realizes dynamic convolution at a global level by using attention mechanism; at the microscopic level however, it still uses a weight-sharing strategy, and may not be good at capturing dynamic correlations.

Time Complexity Analysis. The dynamic weight of SDK is updated in training. For the input MTS $X_{1\sim t}^{1\sim N}$, assume L filters are assigned, and for the lth filter, SDK uses a two-layer convolutional network with kernel size as 1 and padding height as K//2 to assign specific weight to each "element" of $X_{1\sim t}^{1\sim N}$. The initial values of each dynamic weight are assigned by using Xavier strategy, and will be updated during each iteration. The time complexity is: $O((t-1+2*K//2)*(N-1)+(t-1)*(N-1))$. The dynamic weight matrix $W_{1\sim t+(2K-1)//2}^{1\sim N}$ (obtained from formula (1)) is used to make convolution with $X_{1\sim t+(2K-1)//2}^{1\sim N}$, which is an extension of $X_{1\sim t}^{1\sim N}$ and is used to align with $W_{1\sim t+(2K-1)//2}^{1\sim N}$. The convolution kernel size is $K\times N$, so the time complexity is: $O((t+(2K-1)//2-K))$. As seen in formula (2) and (3), $divW$ can divide $W_{1\sim t+(2K-1)//2}^{1\sim N}$ into t $K\times N$ matrices $W^{(K\times N)\times t}$ by assigning kernel size as $K\times N$, padding as 0 and stride as 1. $divX$ can divide $X_{1\sim t}^{1\sim N}$ into t $K\times N$ matrices $X^{(K\times N)\times t}$ by assigning kernel size as $K\times N$, padding as $(2K-1)//2$ and stride as 1.

In formula (4), the output of the lth filter, which is $O_{1\sim t}^{l}$, is the convolution operation between W^{l} and X^{l}. Assume that the gradient of back propagation at $O_{1\sim t}^{l}$ is $\gamma_{1\sim t}^{l}$, then the gradient assigned to each dynamic weight of W^{l} can be expressed as $\nabla(W_i^j) = \sum_z \gamma_z^l \times f'(W_i^j) \times [\partial(O_z^l)/\partial(f(W_i^j))]$, where W_i^j belongs to the zth $W^{K\times N} \in W^{(K\times N)\times t}$ $(z \le t)$, f is the activation function, $i \in [1, t+2(K-1)//2]$ and $j \in [1,N]$. Similarly, the kernel gradient of $conv1$ could be represented as: $\nabla(Wconv1_i^j) = \sum_{i,j} \nabla(W_i^j) \times [\partial(W_i^j)/\partial(f(Wconv1_i^j))] \times f'(Wconv1_i^j)$, where $i \in [1,t]$ and $j \in [1,N]$. In this research, the PyTorch optimizer is adopted to realize the gradient updating automatically. We also visualize the weight change process of SDK, as well as its time consuming in Fig. 4.

3.2.2. Hierarchical Attention (HA) component

The HA component is an integration model of DARNN (Qin et al., 2017) and MLCNN (Cheng et al., 2020). It takes advantages of spatiotemporal attention design from DARNN to obtain spatial (SA) and temporal (TA) attention based representations ht_1, ht_2, ht_3 and ht_4 of input $(O_{1\sim t}^{L})_{t+1}$, $(O_{1\sim t}^{L})_{t+2}$, $(O_{1\sim t}^{L})_{t+3}$ and $(O_{1\sim t}^{L})_{t+4}$ from SDK component.

Spatial Attention model (SA). For multivariate time series prediction, spatial attention could be seen as detecting the attention distributions of all the L columns of $(O_{1\sim t}^{L})_{t+k}$ based on the hidden state of the encoder $hs_k^{\tau} \in R^d$ at time τ $(\tau \in [0,t])$, where $k \in \{1,2,3,4\}$, and d is the size of hidden state. We design LSTM encoder hs_k^{τ} of input $(O_{1\sim t}^{L})_{t+k}^{[\tau,:]}$. The formula could be seen in formula (9):

$$\textbf{Hidden State}: hs_k^{\tau} = os_k^{\tau}(hs_k^{\tau-1}, (O_{1\sim t}^{L})_{t+k}^{[\tau,:]}) \odot tanh(cell\_s_k^{\tau}(hs^{\tau-1}, (O_{1\sim t}^{L})_{t+k}^{[\tau,:]})) \tag{9}$$

where $(O_{1\sim t}^{L})_{t+k}^{[\tau,:]}$ represents the τth $(\tau \le t)$ row of matrix $(O_{1\sim t}^{L})_{t+k}$. os_k^{τ} is the Output Gate, $cell\_s_k^{\tau}$ is the Cell Status, and the output Hidden State is hs_k^{τ}. $tanh$ is the activation function, $\odot$ is the element-wise product. We propose Spatial Attention model, which is inspired by the theory of attention mechanism, to adaptively select the relevant driving series, which is of practical meaning in time series prediction. Given the nth ($n \in L$) input driving (exogenous) series $(O^L_{1\sim t})^{[:,n]}_{t+k}$ for the kth LSTM encoder hs_k at time $\tau \in t$, Spatial Attention model could be constructed via a deterministic attention model by referring to the previous hidden state $hs^{\tau-1}_k$ and the cell status $cell\_s^{\tau-1}_k$ in the encoder LSTM:

$$es^{\tau}_k(n) = (Vs_e)^T \times tanh(Ws_e \times concat[hs^{\tau-1}_k, cell\_s^{\tau-1}_k] + Us_e \times (O^L_{1\sim t})^{[:,n]}_{t+k}) \tag{10}$$

$$\alpha s^{\tau}_k(n) = \frac{exp(es^{\tau}_k(n))}{\sum_{i=1}^{L} exp(es^{\tau}_k(i))} \tag{11}$$

where $concat[hs^{\tau-1}_k, cell\_s^{\tau-1}_k]$ is an operation to concatenate $hs^{\tau-1}_k$ and $cell\_s^{\tau-1}_k$ in a new vector with dimension as $2d$. $Vs_e \in R^t$, $Ws_e \in R^{t\times 2d}$ and $Us_e \in R^{t\times t}$ are weight parameters to learn. $es^{\tau}_k(n)$ is used to measure the importance of the nth input among all the L input of $(O^L_{1\sim t})_{t+k}$ for the kth LSTM encoder at time τ. $\alpha s^{\tau}_k(n)$ is spatial attention weight of the nth input by using a softmax function to normalize $es^{\tau}_k(n)$. The normalization operation could measure the importance of each input time series at different time τ on a unified scale $[0\sim 1]$ and ensure the weight parameters could be jointly trained with other components. Thus for the kth LSTM encoder, the output of spatial attention model at time τ is: $(\widetilde{O}^{1\sim L}_{1\sim t})^{[\tau,n]}_{t+k} = \alpha s^{\tau}_k(n) \times (O^{1\sim L}_{1\sim t})^{[\tau,n]}_{t+k}$, where $n \in N$.

Temporal Attention model with Future Information Fusion (TA-FIF). Temporal attention is mainly used to calculate attention distributions of all input time series along the timeline. As introduced in previous sections, considering the interactions of different future moments could further improve the prediction accuracy (Cheng et al., 2020). Assume at time t, the output of spatial attention model, which is the input of temporal attention model, is $(\widetilde{O}^L_{1\sim t})^{[:,n]}_{t+k}$, where $k \in \{1,2,3,4\}$ and $n \in L$. TA-FIF contains four LSTM layers. For the kth layer LSTM $ht^{k\sim t+k}_k$, $k \in \{1,2,3,4\}$. The expression of hidden state ht^{t+k}_k at time $t+k$ could be seen in formula (12).

$$\textbf{Hidden State}: ht^{t+k}_k = ot^{t+k}_k(ht^{t+k-1}_k, (\widetilde{O}^L_{1\sim t})^{[t,:]}_{t+k}) \odot tanh(cell\_t^{t+k}_k(ht^{t+k-1}_k, (\widetilde{O}^L_{1\sim t})^{[t,:]}_{t+k})) \tag{12}$$

Similar to formula (9) of SA, the hidden state of TA-FIF is based on the same LSTM module and also contains Cell Status $cell\_t^{t+k}_k$, Output Gate ot^{t+k}_k and Hidden State ht^{t+k}_k. The input $(\widetilde{O}^L_{1\sim t})^{[t:,n]}_{t+k}$ is fed into TA-FIF to identify the importance of the nth column. TA-FIF adopts temporal attention to capture influential factors at each time τ for sales predictions from the perspective of timeline angles. The expression of temporal attention could be seen in formula (13).

$$\begin{aligned} ht^{t+k}_k &= \sum_{i=1}^{t+k-1} \alpha t_{ki} \odot ht^i_k \\ \alpha t_{ki} &= \frac{exp\ sim(ht^i_k, ht^{t+k}_k)}{\sum_{j=1}^{t+k-1} exp\ sim(ht^j_k, ht^{t+k}_k)} \end{aligned} \tag{13}$$

The final representation of hidden state ht^{t+k}_k is the weighted sum of all ht^i_k with attention weight as αt_{ki}, where $i \in [1, t+k-1]$. The calculation of αk_{ki} is mainly based on the cosine similarity between ht^i_k and ht^{t+k}_k. The term "future information fusion (FIF)" is similar to existing researches (Cheng et al., 2020; Liu et al., 2021), and could be expressed as that the temporal attention based hidden state at kth layer $ht^{t+k}_k (k < 4)$ is assigned as the initial value of $k+1$th layer ht^{t+k+1}_{k+1}. The main idea of FIF is that we consider the trained model in the kth layer for the prediction task of $\widetilde{Y}^{HA}_{t+k}$ could be applied as prior knowledge for the prediction task of $\widetilde{Y}^{HA}_{t+k+1}$ in the $k+1$th layer. It then draws on the work of MLCNN to adopt a set of functions H_1, H_2, H_3 and H_4 to predict future sales $\widetilde{Y}^{HA}_{t+k}$, $k \in \{1,2,3,4\}$ by fusing forecasting information of different future time $t+k$. The expression is shown in formula:

$$\begin{cases} \widetilde{Y}^{HA}_{t+1} = H_1(ht^{t+1}_1((O^L_{1\sim t})_{t+1})), \\ \widetilde{Y}^{HA}_{t+k} = H_k(ht^{t+k}_k(ht^{t+k-1}_{k-1}, (O^L_{1\sim t})_{t+k})), (2 \le k \le 4) \end{cases} \tag{14}$$

3.2.3. Simultaneous Regression (SR) component

Though the non-linear nature of the SDK and HA improves the capture of non-linear correlations, their abilities to detect stable correlations are reduced. Traditional methods (Cheng et al., 2020; Lai et al., 2018) adopt Auto-regressive (AR) to model the linear correlations, which overlook linear correlations between time series. To address this problem, we replace the AR with SR component with a co-integration test based dynamic programming algorithm (Algorithm 3.2.3).

The co-integration test can find stable correlations within a set of time series, which can reduce the influence of abnormal sequences and, to a degree, resolve the problem of pseudo regression (Box et al., 2015). SR uses dynamic programming to find all the maximum subsets $S = [S_1, S_2, \dots, S_C]$ from input, and all time series in an arbitrary subset S_i of S satisfy co-integration tests with each other. The main purpose of using dynamic programming is to find an efficient way to adopt parallel computing for improving running efficiency. Because a MTS may contain hundreds of different products with the same product type, this phenomenon is particularly common in e-commerce warehouses. Due to the interpreter design of python, threads are hard to be used for significantly improving running efficiency. So we consider adopting python multi-processor. When traditional double-loop strategy is calculated in parallel, due to the complexity of real-time update of global variables between processors, the amount of repeated calculations of each processor is large. Thus, the main idea of dynamic programming is to make co-integration test between a time series with its nearest time series in a MTS instead of traversing all time series. Assume the input window length of SR for capturing correlations of $X^{1\sim N}_{1\sim t}$ is s, so the input N time series could be represented as $X^{1\sim N}_{t-s+1\sim t}$.

Algorithm 1 Co-Integration based Dynamic Programming

```
INPUT: Short-term MTS $X^{1\sim N}_{t-s+1\sim t}(s) = X^{1\sim N}_{t-s+1\sim t}$.
OUTPUT: The Set of subgroups: $S = [S_1, S_2, \dots, S_C]$.
1. $S$ = Function getSubset($X^{1\sim N}_{t-s+1\sim t}$):
2.     Let $X = X^{1\sim N}_{t-s+1\sim t}$;
3.     Define array $pre[i \in N] = i$;
4.     # $pre[N]$ is status vector.
5.     For $(i = 1; i < N; i++)$ :
6.         Assign $pre[i] = i$;
7.         For $(j = i - 1; j >= 0; j--)$ :
8.         # $i$ only needs to find its nearest $j$.
9.             If (ci_ test $(X^i_{t-s+1\sim t}, X^j_{t-s+1\sim t})$ == true):
10.                $pre[i] = j$;
11.                break;
12.     While ($X$ is not empty):
13.     # find all subsets in one loop.
14.         $i = i + 1$.
15.         Define $S_i$ as an empty list;
16.         $k = X$.size() - 1;
17.         $Si.append(X^k_{t-s+1\sim t})$;
18.         While ($k! = pre[k]$):
19.         # Track all the members of $S_i$ from $pre$.
20.             $k = pre[k]$;
21.             $Si.append(X^k_{t-s+1\sim t})$;
22.         $S$.append($S_i$);
23.         $X$.remove($S_i$);
24.     Return $S$;
25.
26. $S$ = getSubset($X^{1\sim N}_{t-s+1\sim t}$);
27. OUTPUT $S$;
```

For a MTS $X^{1\sim N}_{t-s+1\sim t}$ (s is window size), lines 1∼24 describe function getSubset() for obtaining output set S, and line 26 is the main function to iteratively get all the maximum subsets $[S_1, S_2, \dots, S_C]$ from input $X^{1\sim N}_{t-s+1\sim t}$ by invoking function getSubset(). Line 3 defines status vector $pre[N]$. Lines 5∼11 design a strategy to make co-integration test between a time series with its nearest time series instead of traversing all time series. *ci_test* in line 9 is the function to make co-integration test between arbitrary two time series $X^i_{t-s+1\sim t}$ and $X^j_{t-s+1\sim t}$. Line 10 defines status transfer function for updating $pre[N]$. Status function $pre[i] = j$ indicates that $j(j < i)$ is a time series index that has the nearest distance with i among all other time series, which have passed co-integration test with i. Lines 12∼24 describe the strategy to find all subsets in one loop. For a time series $k = X.size() - 1$, status vector $f[k]$ stores the number of other time series in subset S_i when k is assigned to S_i. Status vector $pre[k]$ stores the index of k's previous time series in subset S_i. Thus backward tracking through pre can assign each time series to a specific subset S_i of S in one loop.

The SR component (like the AR component of MLCNN) can predict the values of future multi-step time points ($t+1$, $t+2$, $t+3$, …). Assume that the output S has C subsets, for each subset $S_i (i \in C)$ in S, we use an indicator vector I^{S_i} with length N to note which time series belong to S_i, that is, for each time series $X^j_{t-s+1\sim t}(j < N)$, if $X^j_{t-s+1\sim t}$ belongs to S_i, then $I^{S_i}[j] = 1$, else $I^{S_i}[j] = 0$. The Simultaneous Regression could be described as:

$$\widetilde{Y}^{Si}_{t+k} = \sum_{l=0}^{s} W^{Si}_{k,t-l} \times (I^{Si} \odot X^{1\sim N}_{t-l})^T + b^{Si} \tag{15}$$

where $\widetilde{Y}^{S_i}_{t+k} \in R^N$ represents the predicted sales of N time series in S_i at time $t + k (1 \leq k \leq 4)$. $W^{S_i}_{k,t-l} \in R^{N\times N}$ is the coefficient matrix for predicting sales at time $t+k$. $\odot$ represents the element-wise product. For all S_i in S, the predicted sales $\widetilde{Y}^{SR}_{t+k}$ at $t+1$ could be represented as:

$$\widetilde{Y}^{SR}_{t+k} = \sum_{i=1}^{C} \omega^{Si} \times \widetilde{Y}^{Si}_{t+k} \tag{16}$$

where $\widetilde{Y}^{Si}_{t+k}$ and $\widetilde{Y}^{SR}_{t+k} \in R^N.\omega^{S_i}$ is the learnable weight of each subset, and $\sum_{i=1}^{C} \omega^{Si} = 1$.

Generating the Prediction. We have used the same strategy as earlier studies, such as DARNN, LstNet, MLCNN, etc, adopted weight-sum strategies to combine the results of linear (AR) and non-linear (DNN) component into a unified framework to generate the predictions. As mentioned earlier, for the input multivariate sales time series $X^{1\sim N}_{1\sim t}$, the outputs of HA and SR are $\widetilde{Y}^{HA}_{t+k}$ and $\widetilde{Y}^{SR}_{t+k}$ respectively, where $(1 \leq k \leq 4)$. Then the final predicted sales at time $t + k$ can be represented as:

$$\widetilde{Y} = \alpha \times \widetilde{Y}^{HA}_{t+k} + \beta \times \widetilde{Y}^{SR}_{t+k} \tag{17}$$

where $\alpha, \beta > 0$ are the weight parameters, and $\alpha + \beta = 1$.

3.2.4. Loss function

Absolute Loss (L1-loss). Absolute loss (L1-loss) is widely used for many of the time series forecasting tasks. For a MTS $X^{1\sim N}_{1\sim t}$ in the training data, we define the absolute loss L_{AL} of the proposed model as below in formula (18):

$$L_{AL} = \sum_{k=1}^{4}\sum_{i=1}^{N} |\widetilde{Y}^i_{t+k} - Y^i_{t+k}| \tag{18}$$

where $i \in N$ is the ith time series of $X^{1\sim N}_{1\sim t}$, and k represents the task of predicting the future sales at time $t+k$. Thus, $\widetilde{Y}^i_{t+k}$ and Y^i_{t+k} represent the predicted and true sales of the ith time series at time $t+k$.

Loss of Dynamic Change Detection and Alignment (DC). The DC loss can adjust the prediction results based on captured dynamic changes, and is mainly based on Dynamic Time Warping (DTW) (Sakoe & ChibaLim, 1990). Unlike Mean Average Error (MAE), DTW focuses on finding similar patterns between two time series, and is sensitive to dynamic changes. Assume at time t, the true and predicted ith time series of $X^{1\sim N}_{1\sim t}$ are $Y^i_{t+1\sim t+4}$ and $\widetilde{Y}^i_{t+1\sim t+4}$. The two time series can generate a distance matrix $D^i_{t+1\sim t+4} \in R^{4\times 4}$, where $D_{t+1\sim t+4}(m,n) = |Y^i_{t+n} - \widetilde{Y}^i_{t+m}|$, $i \in N$. DTW can find the minimal distance path from $(1,1)$ to $(4,4)$ of $D^i_{t+1\sim t+4}$ by constructing the optimization association matrix A. The calculation rules of dynamic transfer matrix are defined in follow formula.

$$\begin{aligned} rmax &= D_{t+1\sim t+4}(m,n) + max\{r_1, r_2, r_3\} \\ r_1 &= -A^i[m-1,n-1]/\gamma; r_2 = -A^i[m-1,n]/\gamma; r_3 = -A^i[m,n-1]/\gamma \end{aligned} \tag{19}$$

where $\gamma = 0.01$ is smooth parameter, and $rmax$ indicates the smallest distance around $A^i[m,n]$. Here we define the smooth minimal operator as $smo\_r_s = \frac{1}{Z}exp(r_s - rmax)$, where $Z = \sum_{s=1}^{3} smo\_r_s$ and $s \in [1,2,3]$. The smooth minimal operator is used to design differentiable loss (Cuturi & Blondel, 2017). Then matrix A^i can then be iteratively calculated by follow formula:

$$A^i[m,n] = smo\_r_1 \times A^i[m-1,n-1] + smo\_r_2 \times A^i[m-1,n] + smo\_r_3 \times A^i[m,n-1] \tag{20}$$

Matrix A^i detects dynamic changes from the ith time series. In order to realize dynamic change alignment, we added a time penalty matrix $\Omega^i \in 4\times 4$ (Vallance, Charbonnier, Paul, Dubost, & Blanc, 2017) of the ith time series $\Omega^i[m,n] = (m-n)^2/(4\times 4)$. The purpose of designing Ω^i is for a pair (m,n) with high similarity in A^i, if their distance $\Omega^i[m,n]$ is also high, then a penalty is added to enlarge the current loss for aligning time between $Y^i_{t+1\sim t+4}$ and $\widetilde{Y}^i_{t+1\sim t+4}$. Finally, for all the predicted sales $\widetilde{Y}^i_{t+1\sim t+4} (i \in N)$ of $X^{1\sim N}_{1\sim t}$, the DC loss L_{DC} is defined in follow formula:

$$L_{DC} = \sum_{i=1}^{N} Sum(Mean(A^i \times \Omega^i, dim=0))/(4\times 4) \tag{21}$$

where $Mean(A^i \times \Omega^i, dim=0)$ averages each column $(dim=0)$ of matrix $A^i \times \Omega^i$, and the output is a 1×4 dimension vector. Sum means to sum each element of the output vector. The backward propagation for solving DTW optimization problem can use the Hessian matrix-based gradient decent method (Cuturi & Blondel, 2017).

As introduced above, DC component can provide more future information based on future trend correlations for model training. The premise that DC loss can play a greater role is that the model has a larger parameter space for capturing more pattern candidates. The proposed SDK and HA component can satisfy the condition. SDK provides specific weights for different X^i_{t+j} $(j > 0)$, which can better distinguish the dependent correlation patterns. The spatiotemporal attention of HA can effectively select more useful dependent correlation patterns from the output of SDK. However, the main challenge of capturing future trend correlations is that there are deformations between true and predicted sales. The DC component can overcome this by using an optimized DTW loss with a time penalty matrix.

Loss Function Considering homovariance uncertainty (HU). Here we refer to the loss function proposed by He, Zhu, Wang, Marios, and Zhang (2019), Kendall, Gal, and Cipolla (2018). They consider the homovariance uncertainty (HU) between each loss function. Research has demonstrated its effectiveness in image semantic parsing. We also found that it could significantly improve the performance of sales prediction because it considers the variance differences between different loss functions, and reduces the uncertainty of linear combinations. The final loss function $L_{SDPANet}$ considering homovariance uncertainty is shown in formula (22):

$$L_{SDPANet}(\widetilde{Y}, Y) = \frac{1}{\sigma_1^2} L_{AL}(\widetilde{Y}, Y) + \frac{1}{\sigma_2^2} L_{DC}(\widetilde{Y}, Y) + log\sigma_1\sigma_2 \tag{22}$$

where σ_1 and σ_2 are scalar observational variances of loss function L_{AL} and L_{DC}. The goal of model training is to minimize the loss function of all parameters, which can be achieved by using the stochastic gradient (SGD) optimizer. In this paper, we use Adam algorithm (Das & Ghosh, 2017) to optimize the model parameters.

In summary, each component of SDPANet plays a unique and irreplaceable role in the model, and cooperates with each other. SDK can capture dynamic non-linear correlations between different time series of input $X^{1\sim N}_{1\sim t}$. The output of SDK will be subsequently fed into a HA component to calculate spatiotemporal attention patterns. The dynamic kernel of SDK can expand the parameter space of feature calculation. This helps the HA find more non-linear patterns to better fit the sales time series, which are influenced by complex external environment. SR is a novel component used to detect dynamic linear correlations between time series. The DC component is proposed to address the limitations of existing widely used loss functions. It does this by considering sales correlations between different future time points. Therefore, the DC loss can provide valuable information for the rich dynamic features and patterns of SDK, HA and SR. The SDK, HA and SR components can also realize collaborative training based on the feedback of DC loss.

Table 1
Data description of Galanz and Cainiao.

	Galanz	Cainiao
Product type	190	200
Samples	55,361	74,595
Features	• Product type • Historical sales • Amount of shop discount • Perform discount amount • Discount rate	• Product type • Historical sales • User visits records • Visits to cart • Collections user visits

Table 2
Data description of traffic and exchange-rate.

	Traffic	Exchange-rate
Length of time series	17,544	7,588
Number of variables	862	8
Features	1 h	1 day

4. Experiments

4.1. Experiment data

Experiments are conducted on two datasets: Galanz and Cainiao. Descriptive statistics of both samples and features of the two datasets are summarized in Table 1.

Galanz Dataset: This is collected from one of China's leading home appliance enterprises. Over a two-year period, Galanz stored more than 600 products in 100 warehouses. We selected 190 products and their sales time series records from 10 warehouses.

Cainiao Dataset: The Cainiao Company[1] is one of the largest intelligent logistic companies in China. The data contains sales records of up to 200 products in 5 warehouses.

For both datasets, we used historical time-series to predict sales of each stored product. Each product was grouped, firstly by warehouse, then by type, to generate multivariate time series. For each multivariate time series, training and testing samples were generated by dividing the whole series into a set of sub-series with a minimum length of greater than 24. The last 1∼4 time period of each sub-series was taken as prediction label; other periods were taken as features. This procedure yielded 55,361 samples from the Galanz and 74,595 samples from Cainiao. All datasets were split in chronological order to produce a training set (60%), a validation set (20%), and a test set (20%), named, respectively GWN (Galanz) and CWN (Cainiao). To further assess the practical values of the proposed model, warehouse ids were used to divide Galanz test data into 10 groups (GW1∼GW10) and Cainiao test data into 5 groups (CW1∼CW5).

We also tested the effect of the model on two widely-studied datasets: Traffic and Exchange-Rate (Table 2):

Traffic: 48 months (2015–2016) of hourly data from the California Department of Transportation. The data describes road occupancy rates (between 0 and 1) measured by different sensors on San Francisco Bay area freeways.

Exchange-Rate: A collection consists of the daily exchange rates of eight foreign countries including (Australia, British,UK, Canada, Switzerland, China, Japan, New Zealand and Singapore) ranging from 1990 to 2016.

Each dataset was split into a train set (60%), a validation set (20%) and a test set (20%) in chronological order.

4.2. Metrics

Evaluation is based on the three commonly used metrics defined in Eqs. (23) to (25). Here, for the ith test sample, Y_i^t and $\widetilde{Y}_i^t$ denotes the true and predicted number of sales at time t. RMAE and RRSE are scaled versions of the widely used Mean Absolute Error (MAE) and Root Square Error (RSE), and CORR is the Coefficient of correlation between Y_i^t and $\widetilde{Y}_i^t$. Low values for RMAE, RRSE and a high value of CORR indicates a good performance.

Relative Mean Absolute Error (RMAE):

$$RMAE = \frac{mean(\sum_{(i,t)\in\Omega_{test}} |Y_i^t - \widetilde{Y}_i^t|)}{mean(\sum_{(j,t)\in\Omega_{test}} Y_j^t)} \tag{23}$$

[1] https://tianchi.shuju.aliyun.com/competition/introduction.htm?spm=5176.100068.5678.1.ZBYlCN&raceId=231530.

Root Relative Squared Error (RRSE):

$$RRSE = \frac{\sqrt{\sum_{(i,t)\in\Omega_{test}}(Y_i^t - \widetilde{Y}_i^t)^2}}{\sqrt{\sum_{(i,t)\in\Omega_{test}}(Y_i^t - mean(Y))^2}} \tag{24}$$

Empirical Correlation Coefficient (CORR):

$$CORR = \frac{1}{|\Omega_{test}|} \times \frac{\sum_{(i,t)\in\Omega_{test}}(Y_i^t - mean(Y))(\widetilde{Y}_i^t - mean(\widetilde{Y}))}{\sqrt{\sum_{(i,t)\in\Omega_{test}}(Y_i^t - mean(Y))^2(\widetilde{Y}_i^t - mean(\widetilde{Y}))^2}} \tag{25}$$

4.3. Baselines

- RNN-LSTM: RNN-LSTM is widely used in time series research because its design makes it suitable for processing and predicting events with long intervals and delays in time series (Hochreiter & Schmidhuber, 1997).
- WaveNet[2]: WaveNet (Lee, Jin, Chu, Lim, & Ko, 2021) is mainly based on an extended causal convolutional layer, which allows it to deal with temporal sequences and long-term dependencies without a marked increase in model complexity.
- LSTNet[3]: Long and Short-term Time-series Network (Lai et al., 2018) adopts a CNN-LSTM framework to extract local dependency patterns. It uses convolutional layer and recurrent layer to capture long-term dependency patterns. It proposes a novel recurrent-skip layer to capture periodic properties for forecasting.
- MLCNN[4]: Multi-Level Construal Neural Network (Cheng et al., 2020) is a multi-task deep learning framework that improves predictive performance by fusing information from different future times. The framework uses a Convolutional Neural Network to extract multi-level representations of the raw data, then models interactions between multiple predictive tasks and fuses their future visions through an Encoder–Decoder framework.
- DARNN[5]: The Dual-stage Attention-based RNN (Qin et al., 2017) uses a two-stage attention mechanism for MTS predictions. The first stage learns the weights of input variables through a spatial perspective, and the second stage uses a temporal perspective to learn the weights of hidden states across all time steps.
- MTNet[6]: Memory Time series network (Chang, Sun, Wu, & Shou-De, 2019) is jointly trained by a large memory component, three separate encoders, and an auto-regressive component. The memory component and attention mechanism store long-term correlation patterns through historical data and make prediction results explainable.
- ST-NORM[7]: ST-NORM (Deng, Chen, Jiang, Song, & Tsang, 2021) propose two types of normalization modules including temporal normalization and spatial normalization to refine the high-frequency and local components of the original data respectively. And they can be easily integrated into a canonical deep learning architecture.
- Informer[8]: Informer (Zhou et al., 2021b) designed a probSparse self-attention and distilling operation to address the Transformer's challenges of secondary time complexity and secondary memory usage. At the same time, the carefully designed generative decoder alleviates the limitations of the traditional encoder–decoder architecture.
- Pyraformer[9]: Pyraformer (Liu et al., 2021) can bridge the gap between capturing remote dependencies and achieving low temporal and spatial complexity. And a pyramidal attention Module is introduced, whose inter-scale tree structure combines features of different resolutions, while adjacent connections within the same scale model time dependence of different ranges.
- BHT-ARIMA[10]: BHT-ARIMA (Shi et al., 2020b) is designed to solve practical application problems in MTS prediction. In this model, the source time series data are augmented by high-order tensors with the help of multi-path delay transformation technology, and the classical time series prediction model ARIMA is combined with tensor decomposition. The model has been used to predict PC sales.
- DeepAR[11]: DeepAR (Salinas, Flunkert, Gasthaus, & Januschowski, 2020) is a self-learning prediction algorithm that uses RNN to predict one-dimensional time series. It can learn global models from relevant time series; and can learn complex patterns such as seasonality and data uncertainty growth over time. The model has been used to predict monthly sales of different items sold by a US automobile company.

[2] https://github.com/ibab/tensorflow-wavenet.
[3] https://github.com/laiguokun/LSTNet.
[4] https://github.com/smallGum/MLCNN.
[5] https://github.com/fanyun-sun/DARNN.
[6] https://github.com/Maple728/MTNet.
[7] https://github.com/JLDeng/ST-Norm.
[8] https://github.com/zhouhaoyi/Informer2020.
[9] https://github.com/alipay/Pyraformer.
[10] https://github.com/yokotatsuya/BHT-ARIMA.
[11] https://github.com/arrigonialberto86/deepar.

- Prophet[12]: Prophet (Taylor & Letham, 2018) is an open-source time series prediction framework developed for forecasting at Facebook. Its core function is an additive regression model with three components: a piece-wise logical growth model, a Fourier series based seasonal model, and a vacation impact model based on indicator function and change distribution. The model is widely used in sales predictions.
- StemGNN[13]: Spectral time graph neural network (StemGNN) (Cao et al., 2020) combines Graph Fourier Transform (GFT) and Discrete Fourier Transform (DFT) into an end-to-end framework. GFT and DFT are used to model inter-series correlations and temporal dependencies. The GFT and DFT based spectral representations hold clear patterns and can be predicted effectively by convolution and sequential learning modules.

4.4. Training details

A grid search strategy is conducted for the proposed SDPANet and all baselines for finding the best hyper-parameter settings. For RNN-LSTM, the hidden size is set at 256, full connect layer size is 128, batch size is set at 16. For WaveNet, dense hidden size is set at 64, dilation rates are set at 2^i for i in range [1, 4], Kernel size is set at 3 and the number of filters is 128. For DARNN, the encoder and decoder hidden layer are all set at 128, time step is 28. For LSTNet, CNN and RNN hidden size are set at 100, CNN kernel size is 6, highway window is set at 28. For MTNet, CNN and RNN hidden size are set at 16 and 32 respectively, CNN filter is 3, batch size is 64 and the number of long-term memory is 7. For MLCNN, the number of CNN filters is set at 25, the hidden size of both shared and main LSTM is set at 50, the dropout is 0.2, batch size is 128. For StemGNN, the channel size of each graph convolution layer is set as 64 and the kernel size of CNN is 3, the learning rate is initialized by 0.001 and decayed with rate 0.7 after every 5 epochs. For Informer, the number of heads is set at 8, the dropout is 0.05. For Pyraformer, the number of heads is set at 8, the dropout is 0.05, the number of children of a parent node is 2, the number of adjacent nodes is 3, batch size is 12. For BHT-ARIMA, the tucker decomposition ranks are set to [5, 5], the orthogonality mode is 4 and stop criterion is 0.01. For DeepAR, the learning rate is set at 0.001, the batch size is 16, the dropout is 0.2, the number of RNN hidden units is 32 and the number of RNN hidden layer is 1. For Prophet, the confidence interval is set at 0.8, the change point range is 0.8, the change points number is 1000 and the change point prior scale is 0.05.

For the proposed SDPANet,[14] the filter number of SDK component is chosen from {5, 10, 15, 20, 25, 30, 35, 40, 45, 50}. The maximum input length of time series is chosen from {24, 32, 48, 56, 64}. The RNN hidden state of HA component is chosen from {32, 64, 96, 128, 256}. The learning rate is set as 0.001. Besides, dropout is performed for each neural network layer with its rate set as 0.2. The window size s for SR is assigned as 30. The details of parameter sensitive test could be seen in Section 4.7.

4.5. Main results

Tables 3–5 summarizes the average performance and standard deviations of all methods on Ganalz and Cainiao datasets. The best results are highlighted in bold, and second-best results are underlined. SDPANet outperforms the other baselines on both Galanz and Cainiao test data. The results are summarized below:

Improves on the other 13 baselines. For Galanz GWN test data, the average improvements in RMAE and RRSE are about 53% and 42% respectively, while for Cainiao CWN data the improvements are about 30% and 37%. Compared with the best baselines, the improvements are 20% and 6.5% on Galanz and 6.3% and 6.6% on Cainiao in terms of RMAE and RRSE. SDPANet achieves the best performances in 7 of Galanz's 10 warehouses, and in 3 of Cainiao's 5 warehouses in terms of both RMAE and RRSE.

Compared with the three sales prediction models, which are BHT-ARIMA, DeepAR and Prophet, SDPANet significantly outperforms the best of the three baselines on GWN and CWN. For RMAE, RRSE and CORR the improvements are, respectively, 43%, 4%, 4% (on Galanz), and 18.1%, 23.1%, 3.7% (on Cainiao). The model also improves significantly on data from individual warehouses. The competitive baselines, such as BHT-ARIMA converts the input $X_{1\sim t}^{1\sim N}$ into a higher order tensor by conducting duplication matrix operations; it then calculates the kernel tensor by using low-rank tensor decomposition. The kernel tensor is used to predict future sales. Unlike BHT-ARIMA, the proposed SDPANet adopts SDK giving more flexibility in the assignment of dynamic weight. SDPANet combines dynamic weight with the attention-based sequence model of HA, which can better extract different patterns from different time periods. BHT-ARIMA maps the internal correlations of the input MTS to each orthogonal dimension of the kernel tensor without particularly modeling the dynamic correlations. In addition, SDPANet treats linear and non-linear correlations separately, and the DC loss provides more future trend information for better model training.

Higher degree of correlations. Values of CORR for baseline models were relatively small, indicating there exists large deviations between true and predicted values at some prediction time points. As for GWN and CWN, the improvement of CORR compared with the best baselines are 2.1% and 36% respectively. The average CORR of all baselines for GWN and CWN was, respectively, 0.47 and 0.30. These values are far smaller than the averages of 0.93 and 0.80 gained using SDPANet. This indicates that the proposed SDPANet can significantly reduce the predictions with large deviations of both Galanz and Cainiao, and the prediction results are more stable.

[12] https://github.com/facebook/prophet.

[13] https://github.com/microsoft/StemGNN.

[14] https://github.com/inksyy/SDPANet.

Table 3
Performance on Galanz dataset (GW = Galanz Warehouse, GW1~GW5 are the top 5 ranked warehouses).

Galanz							
Metrics	Method	GW1	GW2	GW3	GW4	GW5	GWN
RMAE	LSTM	0.77 ± 0.005	0.59 ± 0.005	0.61 ± 0.042	1.18 ± 0.000	0.78 ± 0.013	1.21 ± 0.011
	WaveNet	0.56 ± 0.057	0.06 ± 0.001	0.54 ± 0.033	0.83 ± 0.076	0.82 ± 0.008	1.20 ± 0.023
	MLCNN	0.20 ± 0.001	0.16 ± 0.004	0.24 ± 0.13	0.35 ± 0.003	0.45 ± 0.020	0.65 ± 0.006
	LSTNet	0.51 ± 0.002	0.06 ± 0.003	0.74 ± 0.005	0.76 ± 0.046	0.85 ± 0.032	1.18 ± 0.014
	MTNet	0.52 ± 0.011	0.18 ± 0.004	0.71 ± 0.003	0.79 ± 0.005	0.87 ± 0.004	1.24 ± 0.008
	DARNN	0.71 ± 0.006	0.57 ± 0.038	0.22 ± 0.004	0.82 ± 0.078	0.73 ± 0.028	1.25 ± 0.026
	StemGNN	0.51 ± 0.000	0.05 ± 0.008	0.74 ± 0.001	0.76 ± 0.023	0.85 ± 0.008	1.10 ± 0.007
	Informer	0.51 ± 0.001	0.07 ± 0.001	0.75 ± 0.000	0.76 ± 0.006	0.82 ± 0.063	1.14 ± 0.007
	Pyraformer	0.51 ± 0.003	0.10 ± 0.003	0.76 ± 0.004	0.76 ± 0.000	0.88 ± 0.004	1.21 ± 0.003
	BHT-ARIMA	0.25 ± 0.000	0.06 ± 0.000	0.53 ± 0.000	0.36 ± 0.000	0.64 ± 0.001	0.95 ± 0.000
	ST-NORM	0.23 ± 0.043	0.08 ± 0.007	0.20 ± 0.007	**0.12 ± 0.023**	0.69 ± 0.015	0.83 ± 0.004
	DeepAR	0.52 ± 0.000	0.25 ± 0.001	0.77 ± 0.000	0.77 ± 0.000	0.86 ± 0.003	1.21.±0.001
	prophet	0.51 ± 0.000	1.07 ± 0.000	0.43 ± 0.000	0.36 ± 0.000	1.02 ± 0.000	1.14 ± 0.000
	SDPANet	**0.13 ± 0.039**	**0.05 ± 0.002**	**0.16 ± 0.003**	0.16 ± 0.011	**0.33 ± 0.009**	**0.52 ± 0.006**
RRSE	LSTM	1.62 ± 0.076	13.2 ± 0.002	0.85 ± 0.001	1.73 ± 0.009	0.86 ± 0.005	1.03 ± 0.031
	WaveNet	1.01 ± 0.008	1.06 ± 0.012	1.03 ± 0.003	1.01 ± 0.013	1.10 ± 0.011	1.01 ± 0.003
	MLCNN	0.65 ± 0.065	2.37 ± 0.299	0.22 ± 0.007	0.14 ± 0.015	0.55 ± 0.001	1.24 ± 0.019
	LSTNet	1.03 ± 0.012	1.16 ± 0.019	1.03 ± 0.012	1.02 ± 0.011	1.12 ± 0.006	1.03 ± 0.014
	MTNet	1.05 ± 0.006	2.77 ± 0.099	1.04 ± 0.06	1.12 ± 0.083	1.19 ± 0.064	1.02 ± 0.043
	DARNN	0.82 ± 0.014	3.56 ± 0.034	0.23 ± 0.013	0.85 ± 0.004	0.71 ± 0.045	0.68 ± 0.024
	StemGNN	1.04 ± 0.001	1.05 ± 0.148	1.04 ± 0.004	1.03 ± 0.002	1.12 ± 0.002	1.29 ± 0.003
	Informer	1.02 ± 0.005	1.06 ± 0.022	1.01 ± 0.003	1.01 ± 0.002	1.11 ± 0.002	1.03 ± 0.001
	Pyraformer	1.01 ± 0.000	1.52 ± 0.088	1.00 ± 0.000	1.00 ± 0.002	1.09 ± 0.004	1.03 ± 0.002
	BHT-ARIMA	0.44 ± 0.000	1.18 ± 0.000	0.71 ± 0.000	0.41 ± 0.000	0.84 ± 0.000	1.49 ± 0.000
	ST-NORM	0.72 ± 0.015	1.21 ± 0.061	0.18 ± 0.002	**0.09 ± 0.028**	0.75 ± 0.009	0.64 ± 0.009
	DeepAR	1.01 ± 0.000	2.96 ± 0.015	1.01 ± 0.001	1.01 ± 0.000	1.09 ± 0.002	1.03 ± 0.000
	prophet	0.76 ± 0.000	32.11 ± 0.000	0.35 ± 0.000	0.59 ± 0.000	0.34 ± 0.012	0.62 ± 0.000
	SDPANet	**0.15 ± 0.009**	**0.75 ± 0.028**	**0.17 ± 0.003**	0.13 ± 0.007	**0.33 ± 0.001**	**0.58 ± 0.011**
CORR	LSTM	0.99 ± 0.001	0.41 ± 0.000	0.99 ± 0.001	0.99 ± 0.002	0.73 ± 0.008	0.30 ± 0.001
	WaveNet	0.38 ± 0.019	0.24 ± 0.140	0.23 ± 0.023	0.50 ± 0.076	0.59 ± 0.058	0.28 ± 0.007
	MLCNN	0.94 ± 0.006	0.68 ± 0.148	**1.00 ± 0.001**	0.96 ± 0.005	0.94 ± 0.004	0.59 ± 0.007
	LSTNet	0.08 ± 0.002	0.38 ± 0.012	0.17 ± 0.010	0.02 ± 0.026	0.01 ± 0.046	0.02 ± 0.003
	MTNet	0.05 ± 0.014	0.05 ± 0.003	0.23 ± 0.005	0.03 ± 0.026	0.12 ± 0.003	0.12 ± 0.011
	DARNN	0.70 ± 0.004	0.20 ± 0.089	0.99 ± 0.001	0.77 ± 0.046	0.76 ± 0.011	0.74 ± 0.042
	StemGNN	0.95 ± 0.018	0.39 ± 0.100	0.96 ± 0.001	0.92 ± 0.024	0.65 ± 0.043	0.14 ± 0.013
	Informer	0.74 ± 0.186	0.31 ± 0.093	0.82 ± 0.105	0.86 ± 0.002	0.61 ± 0.051	0.53 ± 0.029
	Pyraformer	0.79 ± 0.019	0.42 ± 0.029	0.86 ± 0.039	0.80 ± 0.086	0.50 ± 0.095	0.57 ± 0.004
	BHT-ARIMA	0.95 ± 0.000	0.16 ± 0.000	0.94 ± 0.000	0.95 ± 0.000	0.80 ± 0.003	0.29 ± 0.000
	ST-NORM	0.93 ± 0.014	0.17 ± 0.043	0.94 ± 0.001	0.95 ± 0.002	0.80 ± 0.155	0.78 ± 0.009
	DeepAR	0.28 ± 0.032	0.32 ± 0.062	0.13 ± 0.166	0.36 ± 0.035	0.36 ± 0.047	0.08 ± 0.018
	prophet	0.94 ± 0.000	0.20 ± 0.000	0.94 ± 0.000	0.78 ± 0.000	0.78 ± 0.000	0.89 ± 0.000
	SDPANet	**0.99 ± 0.000**	**0.73 ± 0.012**	**1.00 ± 0.001**	**1.00 ± 0.002**	**0.95 ± 0.006**	**0.93 ± 0.002**

The performances of the proposed SDPANet on Traffic and Exchange-Rate data are shown in Tables 6 and 7. Horizon represents different training windows, and is assigned values of 3, 6, 12, 24 respectively for both Traffic and Exchange-Rate. Performances of AR, RNN-GRU, LSTNet and MTNet are reported in Chang et al. (2019), Cheng et al. (2020), Lai et al. (2018). The three most competitive baselines (MLCNN, StemGNN and ST-Norm) are also selected. 10-folder cross validation is used to make evaluations. The Traffic time series has a strong periodic pattern, while the time series of different exchange rates has strong correlation patterns; therefore, the effects of all models are relatively close. However, SDPANet still outperforms most of the baselines on both Traffic and Exchange-Rate. For RSE and CORR the average improvements are, respectively, 11% and 10% (Traffic), and 1.5% and 12% (Exchange-Rate). When compared with the best baselines, the average improvements are 2.7% and 0.5% for RSE and CORR respectively.

We display a few qualitative examples based on selected products to further illustrate the performance of the proposed model on large sales predictions and future $t + k$ time periods sales predictions. Experimental results are summarized in Figs. 5 and 6. In Fig. 5, SDPANet could obtain the best performance in predicting large sales on 22 selected products compared with baselines. Take product ID 2 as an example (The curve is marked by red rectangle), the predicted value 1968 is very close to the true value 1865. Though the results of StemGNN, MLCNN and DARNN are competitive, the errors are still large compared with SPDANet. It seems that LSTNet and MTNet could not effectively capture useful patterns for sales predictions.

Fig. 6 shows two examples from both Galanz and Cainiao for performance evaluation on future $t + k$ time periods predictions. Experimental results clearly exhibit that the proposed SDPANet outperformances all the baselines for fusing and predicting different future visions. The state-of-the-art methods could not effectively learn useful knowledge from train data to capture dynamic changes in test data. While the proposed SDPANet could in one aspect obtain more accurate predictions at each future time point, and in

Table 4
Performance on Galanz dataset (GW = Galanz Warehouse, GW6~GW10 are the last 5 ranked warehouses).

Galanz							
Metrics	Method	GW6	GW7	GW8	GW9	GW10	GWN
RMAE	LSTM	0.19 ± 0.017	1.25 ± 0.126	0.85 ± 0.002	0.98 ± 0.008	1.18 ± 0.177	1.21 ± 0.011
	WaveNet	1.14 ± 0.021	1.05 ± 0.098	1.38 ± 0.114	1.19 ± 0.239	1.24 ± 0.004	1.20 ± 0.023
	MLCNN	0.15 ± 0.013	0.75 ± 0.025	0.60 ± 0.030	0.62 ± 0.048	0.52 ± 0.010	0.65 ± 0.006
	LSTNet	0.04 ± 0.027	1.08 ± 0.035	0.93 ± 0.097	0.98 ± 0.008	1.05 ± 0.020	1.18 ± 0.014
	MTNet	0.29 ± 0.011	0.95 ± 0.001	1.25 ± 0.023	1.12 ± 0.006	1.37 ± 0.004	1.24 ± 0.008
	DARNN	0.29 ± 0.032	1.15 ± 0.068	1.08 ± 0.108	0.88 ± 0.056	0.98 ± 0.021	1.25 ± 0.026
	StemGNN	0.04 ± 0.003	0.93 ± 0.013	1.25 ± 0.015	0.98 ± 0.008	1.23 ± 0.011	1.10 ± 0.007
	Informer	0.47 ± 0.042	**0.51 ± 0.046**	1.10 ± 0.123	1.12 ± 0.096	1.16 ± 0.135	1.14 ± 0.007
	Pyraformer	0.05 ± 0.003	0.98 ± 0.002	1.23 ± 0.001	1.03 ± 0.002	1.28 ± 0.004	1.21 ± 0.003
	BHT-ARIMA	**0.03 ± 0.000**	0.79 ± 0.000	1.06 ± 0.003	0.81 ± 0.000	1.00 ± 0.000	0.95 ± 0.000
	ST-NORM	0.08 ± 0.009	0.74 ± 0.011	0.85 ± 0.015	0.76 ± 0.006	0.94 ± 0.051	0.83 ± 0.004
	DeepAR	0.08 ± 0.002	0.96 ± 0.000	1.20 ± 0.001	1.01 ± 0.001	1.27 ± 0.001	1.21 ± 0.001
	prophet	0.34 ± 0.000	1.12 ± 0.000	1.03 ± 0.000	0.87 ± 0.000	0.72 ± 0.000	1.14 ± 0.000
	SDPANet	0.09 ± 0.012	0.60 ± 0.005	**0.54 ± 0.021**	**0.46 ± 0.011**	**0.37 ± 0.007**	**0.52 ± 0.006**
RRSE	LSTM	4.17 ± 0.020	0.80 ± 0.000	0.59 ± 0.006	0.87 ± 0.006	0.50 ± 0.000	1.03 ± 0.031
	WaveNet	0.98 ± 0.001	1.01 ± 0.034	0.94 ± 0.003	0.98 ± 0.010	1.05 ± 0.001	1.01 ± 0.003
	MLCNN	2.61 ± 0.009	1.87 ± 0.136	0.83 ± 0.003	0.65 ± 0.019	0.47 ± 0.022	1.24 ± 0.019
	LSTNet	1.03 ± 0.023	1.04 ± 0.003	1.09 ± 0.031	1.07 ± 0.004	1.10 ± 0.043	1.03 ± 0.014
	MTNet	1.08 ± 0.033	1.15 ± 0.105	1.11 ± 0.003	1.08 ± 0.012	1.14 ± 0.109	1.02 ± 0.043
	DARNN	5.16 ± 0.012	0.88 ± 0.015	0.74 ± 0.001	0.74 ± 0.044	0.84 ± 0.011	0.68 ± 0.024
	StemGNN	1.05 ± 0.080	1.04 ± 0.000	1.12 ± 0.000	1.07 ± 0.001	1.05 ± 0.003	1.29 ± 0.003
	Informer	1.04 ± 0.010	1.04 ± 0.000	1.10 ± 0.000	1.06 ± 0.001	1.04 ± 0.001	1.03 ± 0.001
	Pyraformer	1.00 ± 0.000	1.04 ± 0.000	1.09 ± 0.004	1.06 ± 0.000	1.04 ± 0.000	1.03 ± 0.002
	BHT-ARIMA	**0.63 ± 0.000**	0.91 ± 0.000	0.91 ± 0.001	0.85 ± 0.000	0.79 ± 0.000	1.49 ± 0.000
	ST-NORM	0.99 ± 0.002	0.82 ± 0.018	0.70 ± 0.020	0.68 ± 0.012	0.74 ± 0.040	0.64 ± 0.009
	DeepAR	0.97 ± 0.008	1.04 ± 0.000	1.09 ± 0.001	1.05 ± 0.000	1.04 ± 0.000	1.03 ± 0.000
	prophet	7.80 ± 0.000	0.81 ± 0.000	0.93 ± 0.000	0.59 ± 0.000	0.33 ± 0.000	0.62 ± 0.000
	SDPANet	0.73 ± 0.013	**0.66 ± 0.021**	**0.51 ± 0.004**	**0.40 ± 0.007**	**0.18 ± 0.004**	**0.58 ± 0.011**
CORR	LSTM	**0.95 ± 0.002**	0.64 ± 0.002	0.86 ± 0.008	0.76 ± 0.002	0.94 ± 0.009	0.30 ± 0.001
	WaveNet	0.29 ± 0.004	0.08 ± 0.005	0.51 ± 0.061	0.33 ± 0.012	0.018 ± 0.010	0.28 ± 0.007
	MLCNN	0.86 ± 0.007	0.64 ± 0.012	0.87 ± 0.002	0.91 ± 0.004	0.93 ± 0.064	0.59 ± 0.007
	LSTNet	0.17 ± 0.012	0.05 ± 0.023	0.02 ± 0.065	0.01 ± 0.002	0.101 ± 0.090	0.02 ± 0.003
	MTNet	0.21 ± 0.028	0.07 ± 0.030	0.01 ± 0.005	0.03 ± 0.019	0.18 ± 0.012	0.12 ± 0.011
	DARNN	0.86 ± 0.004	0.52 ± 0.016	0.77 ± 0.033	0.67 ± 0.053	0.57 ± 0.016	0.74 ± 0.042
	StemGNN	0.88 ± 0.003	0.61 ± 0.000	0.72 ± 0.000	0.72 ± 0.024	0.77 ± 0.018	0.14 ± 0.013
	Informer	0.23 ± 0.116	0.49 ± 0.003	0.71 ± 0.051	0.58 ± 0.139	0.55 ± 0.040	0.53 ± 0.029
	Pyraformer	0.36 ± 0.018	0.41 ± 0.007	0.50 ± 0.029	0.55 ± 0.115	0.74 ± 0.001	0.57 ± 0.004
	BHT-ARIMA	0.88 ± 0.000	0.62 ± 0.000	0.67 ± 0.000	0.65 ± 0.001	0.87 ± 0.000	0.29 ± 0.000
	ST-NORM	0.57 ± 0.013	0.60 ± 0.024	0.70 ± 0.021	0.72 ± 0.006	0.74 ± 0.094	0.78 ± 0.009
	DeepAR	0.33 ± 0.051	0.24 ± 0.137	0.40 ± 0.009	0.42 ± 0.016	0.23 ± 0.122	0.08 ± 0.002
	prophet	0.76 ± 0.000	0.76 ± 0.000	0.61 ± 0.000	0.88 ± 0.000	0.95 ± 0.000	0.89 ± 0.000
	SDPANet	0.92 ± 0.004	**0.84 ± 0.012**	**0.88 ± 0.016**	**0.95 ± 0.001**	**0.98 ± 0.002**	**0.93 ± 0.002**

another aspect ensure the shape and trend correlations between true and predicted values to the maximum extent (Higher value of CORR as seen in Fig. 6).

4.6. *Ablation test*

To demonstrate the effectiveness of each model component, we compare SDPANet with 9 variants as follows.

- **SDPANet-SDK**: We remove the SDK module and replace it with the CNN module.
- **SDPANet-HA**: We replace the Hierarchical Attention (HA) Component with shared-main LSTM of MLCNN.
- **SDPANet-TA**: We replace the Temporal Attention(TA) Component with shared-main LSTM of MLCNN.
- **SDPANet-SA**: We remove the Spatial Attention(SA) Component.
- **SDPANet-DC**: We remove the dynamic change (DC) loss function and replace it with the L1 loss function.
- **SDPANet-HU**: We replace the Homovariance Uncertainty (HU) loss with approximate optimal weight.
- **SDPANet-SR**: We remove the Simultaneous Regression (SR) component.
- **SDPANet-SR+AR**: We replace the Simultaneous Regression (SR) with Auto Regression (AR) component.
- **SDPANet-SDK+DCA**: We replace the Dynamic Kernel (SDK) component with Dynamic Convolution with Attention (DCA) (Chen et al., 2020).

Table 5
Performance on Cainiao dataset (CW = Cainiao Warehouse, CW1∼CW5 are the 5 warehouses).

Cainiao							
Metrics	Method	CW1	CW2	CW3	CW4	CW5	GWN
RMAE	LSTM	1.192 ± 0.009	1.247 ± 0.039	0.940 ± 0.001	1.691 ± 0.049	0.939 ± 0.004	1.170 ± 0.001
	WaveNet	1.364 ± 0.007	1.391 ± 0.000	1.053 ± 0.009	1.882 ± 0.022	1.135 ± 0.049	1.340 ± 0.002
	MLCNN	1.198 ± 0.007	1.165 ± 0.000	0.994 ± 0.007	1.819 ± 0.008	1.016 ± 0.022	1.218 ± 0.001
	LSTNet	1.229 ± 0.003	1.109 ± 0.005	1.004 ± 0.001	1.814 ± 0.003	0.982 ± 0.004	1.159 ± 0.006
	MTNet	1.395 ± 0.004	1.255 ± 0.004	1.304 ± 0.003	2.274 ± 0.003	1.164 ± 0.004	1.461 ± 0.006
	DARNN	1.197 ± 0.016	1.247 ± 0.089	1.315 ± 0.081	2.064 ± 0.098	1.343 ± 0.090	1.479 ± 0.043
	StemGNN	1.090 ± 0.234	0.789 ± 0.143	0.881 ± 0.282	1.569 ± 0.328	0.810 ± 0.249	0.988 ± 0.215
	Informer	1.329 ± 0.002	1.244 ± 0.000	1.056 ± 0.000	1.854 ± 0.047	1.063 ± 0.003	1.302 ± 0.001
	Pyraformer	1.335 ± 0.000	1.244 ± 0.000	1.056 ± 0.001	1.907 ± 0.001	1.068 ± 0.000	1.307 ± 0.001
	BHT-ARIMA	1.271 ± 0.000	1.188 ± 0.000	1.008 ± 0.000	1.796 ± 0.001	1.014 ± 0.000	1.241 ± 0.000
	ST-NORM	0.758 ± 0.097	**0.638 ± 0.007**	0.591 ± 0.007	1.254 ± 0.012	0.704 ± 0.143	0.788 ± 0.015
	DeepAR	1.012 ± 0.022	1.006 ± 0.062	0.837 ± 0.027	1.511 ± 0.009	0.826 ± 0.001	1.026 ± 0.001
	prophet	0.852 ± 0.000	0.879 ± 0.000	0.700 ± 0.000	1.248 ± 0.000	0.950 ± 0.000	0.918 ± 0.000
	SDPANet	**0.751 ± 0.006**	0.707 ± 0.004	**0.587 ± 0.012**	**1.120 ± 0.009**	**0.572 ± 0.005**	**0.738 ± 0.006**
RRSE	LSTM	1.121 ± 0.028	1.216 ± 0.013	1.307 ± 0.032	1.026 ± 0.010	1.014 ± 0.005	1.017 ± 0.004
	WaveNet	1.875 ± 0.022	1.335 ± 0.033	1.977 ± 0.016	2.279 ± 0.026	1.165 ± 0.015	1.427 ± 0.021
	MLCNN	0.951 ± 0.008	1.048 ± 0.039	1.015 ± 0.000	0.990 ± 0.003	0.982 ± 0.004	0.988 ± 0.001
	LSTNet	1.000 ± 0.001	0.969 ± 0.002	1.036 ± 0.001	1.003 ± 0.001	0.991 ± 0.002	0.997 ± 0.003
	MTNet	1.094 ± 0.005	1.132 ± 0.006	1.531 ± 0.001	1.346 ± 0.002	1.037 ± 0.004	1.093 ± 0.006
	DARNN	0.897 ± 0.004	0.976 ± 0.029	0.952 ± 0.062	1.012 ± 0.011	1.005 ± 0.001	0.990 ± 0.010
	StemGNN	0.847 ± 0.197	0.916 ± 0.284	2.792 ± 2.090	0.909 ± 0.120	0.804 ± 0.229	0.751 ± 0.031
	Informer	1.043 ± 0.001	1.040 ± 0.001	1.049 ± 0.000	1.029 ± 0.000	1.032 ± 0.001	1.032 ± 0.000
	Pyraformer	1.042 ± 0.001	1.038 ± 0.001	1.047 ± 0.000	1.029 ± 0.000	1.032 ± 0.000	1.032 ± 0.000
	BHT-ARIMA	0.978 ± 0.000	1.019 ± 0.000	0.998 ± 0.000	0.936 ± 0.000	0.985 ± 0.000	0.961 ± 0.000
	ST-NORM	**0.646 ± 0.035**	**0.619 ± 0.012**	0.673 ± 0.028	0.770 ± 0.018	0.650 ± 0.067	0.732 ± 0.008
	DeepAR	0.859 ± 0.016	0.878 ± 0.014	0.899 ± 0.017	0.946 ± 0.001	0.892 ± 0.008	0.915 ± 0.004
	prophet	1.110 ± 0.000	1.056 ± 0.000	1.113 ± 0.000	0.980 ± 0.000	1.317 ± 0.000	1.078 ± 0.000
	SDPANet	0.720 ± 0.006	0.707 ± 0.022	**0.654 ± 0.002**	**0.693 ± 0.031**	**0.641 ± 0.027**	**0.684 ± 0.009**
CORR	LSTM	0.328 ± 0.007	0.327 ± 0.010	0.361 ± 0.001	0.261 ± 0.003	0.278 ± 0.002	0.287 ± 0.001
	WaveNet	0.231 ± 0.024	0.262 ± 0.003	0.246 ± 0.001	0.184 ± 0.004	0.212 ± 0.001	0.087 ± 0.050
	MLCNN	**0.826 ± 0.014**	0.743 ± 0.003	0.634 ± 0.002	0.803 ± 0.008	0.806 ± 0.010	0.713 ± 0.008
	LSTNet	0.278 ± 0.003	0.337 ± 0.005	0.161 ± 0.000	0.216 ± 0.001	0.271 ± 0.008	0.246 ± 0.001
	MTNet	0.012 ± 0.001	0.031 ± 0.002	0.005 ± 0.001	0.004 ± 0.001	0.029 ± 0.001	0.006 ± 0.001
	DARNN	0.668 ± 0.010	0.557 ± 0.026	0.617 ± 0.020	0.615 ± 0.003	**0.809 ± 0.012**	0.192 ± 0.011
	StemGNN	0.741 ± 0.013	0.713 ± 0.071	0.796 ± 0.051	0.790 ± 0.026	0.792 ± 0.028	0.509 ± 0.233
	Informer	0.182 ± 0.066	0.177 ± 0.070	0.150 ± 0.013	0.199 ± 0.026	0.251 ± 0.081	0.185 ± 0.009
	Pyraformer	0.195 ± 0.002	0.114 ± 0.000	0.146 ± 0.040	0.129 ± 0.032	0.178 ± 0.029	0.139 ± 0.003
	BHT-ARIMA	0.681 ± 0.000	0.415 ± 0.000	0.676 ± 0.000	0.800 ± 0.003	0.618 ± 0.000	0.721 ± 0.002
	ST-NORM	0.792 ± 0.037	0.776 ± 0.010	0.771 ± 0.017	0.743 ± 0.020	0.765 ± 0.055	0.739 ± 0.003
	DeepAR	0.710 ± 0.018	0.645 ± 0.014	0.663 ± 0.024	0.581 ± 0.000	0.631 ± 0.008	0.610 ± 0.004
	prophet	0.646 ± 0.000	0.676 ± 0.000	0.639 ± 0.000	0.587 ± 0.000	0.763 ± 0.000	0.636 ± 0.000
	SDPANet	0.722 ± 0.027	**0.783 ± 0.005**	**0.805 ± 0.001**	**0.808 ± 0.004**	0.793 ± 0.019	**0.758 ± 0.006**

For all the variants, we tune their hidden dimension to make them have similar numbers of model parameters to the completed SDPANet model, eliminating the influences of different model complexity. Fig. 7 presents the results of comparison. Important observations from these results are listed as follows.

Removing the SR component (SDPANet-SR) from SDPANet causes the most significant performance drops on both datasets in terms of all metrics. Compared with traditional AR components (SDPANet-SR+AR), the proposed SR could also improve RMAE, RRSE and CORR by 5.8%, 9.1% and 7.3% on average on both Galanz and Cainiao. Compared with Shared-Main LSTM of MLCNN, the proposed HA (HA contains both ST and TA) could obtain significant improvement. The average improvement of RMAE, RRSE and CORR are 3.9%, 7.3% and 4.5% on average (SDPANet-HA). DC loss also provides significant contributions towards model performance. The improvements of RRSE and CORR are particularly obvious, which are 7.3%, 14.1% and 18.5%. DC loss considering HU also contributes positive improvement towards model performance. Removing the SDK component (SDPANet-SDK) causes significant performance drops, which indicates using kernel-specific methods could effectively capture dynamic changing patterns from MTS. In addition, compared with Dynamic Convolution (SDPANet-SDK+DCA), the proposed model can also have obvious improvements, because SDK focuses on capturing correlations from more micro level.

A case study of illustrating working mechanism of SR component could be seen in Fig. 8. For a MTS with 6 sales time series $X_1 \sim X_6$, Augmented Dickey–Fuller (ADF) test is proposed firstly to check whether each time series is stable. In this study, all sales time series pass ADF test with high confidence, and p value of each time series is smaller than 0.05. Then co-integration based dynamic programming (3.2.3) of SR is proposed to group the 6 time series into different subsets. The main advantage of adopting dynamic programming is that it only needs to make co-integration test between two adjacent time series (ex. X_2 only needs to make co-integration test with X_3). Three subsets $S_1(X_1, X_4)$, $S_2(X_2, X_3, X_5)$ and $S_3(X_6)$ are detected. Time series in each subset have stable linear correlation patterns with each other, which could be calculated by using simultaneous equation.

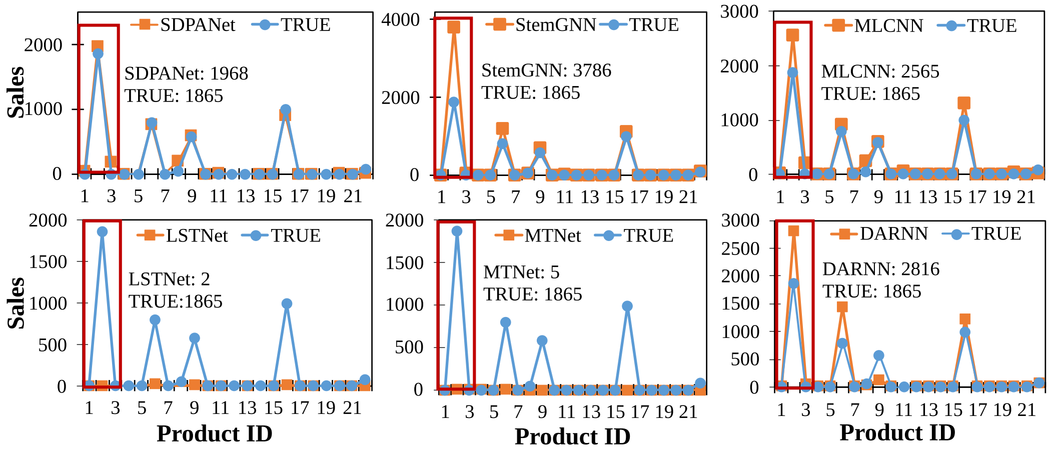


Fig. 5. The performance of the proposed model on selected 22 products.

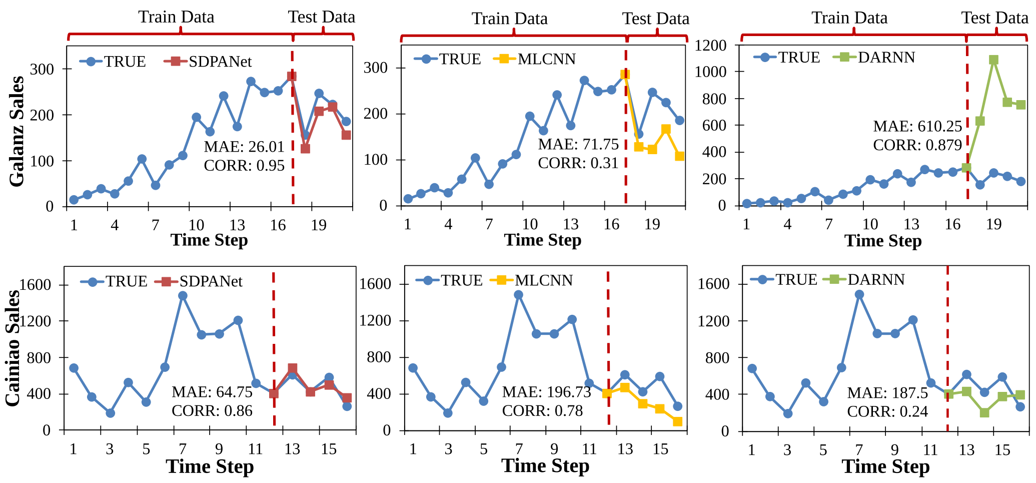


Fig. 6. The performance of the proposed model on selected products for $t + k$ predictions.

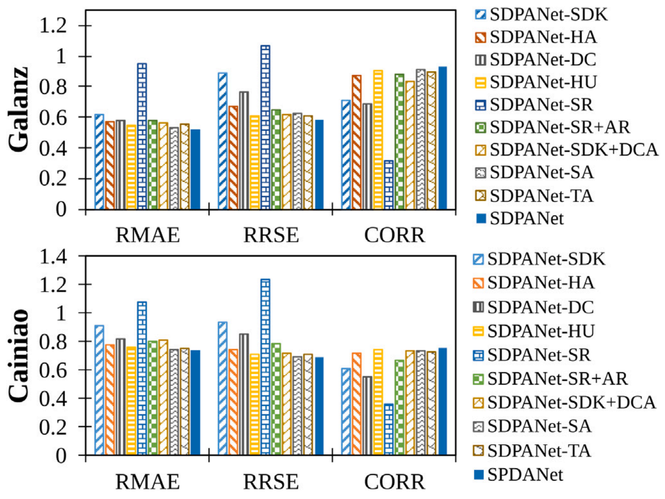


Fig. 7. Results of variant comparison.

Table 6
Performance of the proposed SDPANet model on Traffic dataset.

Dataset		Traffic			
		Horizon			
Methods	Metrics	3	6	12	24
AR[a]	RSE	0.5991	0.6218	0.6252	0.6293
	CORR	0.7752	0.7568	0.7544	0.7519
RNN-GRU[a]	RSE	0.5358	0.5522	0.5562	0.5633
	CORR	0.8511	0.8405	0.8345	0.8300
LSTNet-Skip[a]	RSE	0.4777	0.4893	0.4950	0.4973
	CORR	0.8721	0.8690	0.8614	0.8588
LSTNet-Attn[a]	RSE	0.4897	0.4973	0.5173	0.5300
	CORR	0.8704	0.8669	0.8540	0.8429
MTNet[a]	RSE	0.4764	0.4855	0.4877	0.5023
	CORR	0.8728	0.8681	**0.8644**	0.8570
ST-Norm[b]	RSE	0.9870 ± 0.0080	1.0320 ± 0.0010	0.9465 ± 0.0065	0.9770 ± 0.0080
	CORR	0.3785 ± 0.0605	0.3515 ± 0.0175	0.3140 ± 0.0130	0.1160 ± 0.0120
MLCNN[b]	RSE	0.4723 ± 0.0005	0.4808 ± 0.0015	0.4853 ± 0.0016	0.5129 ± 0.0020
	CORR	0.8728 ± 0.0007	0.8688 ± 0.0013	0.8630 ± 0.0007	0.8448 ± 0.0003
SDPANet	RSE	**0.4716 ± 0.0008**	**0.4803 ± 0.0005**	**0.4845 ± 0.0008**	**0.4959 ± 0.0047**
	CORR	**0.8730 ± 0.0002**	**0.8693 ± 0.0011**	0.8632 ± 0.0008	**0.8578 ± 0.0034**

[a]The results are retrieved from Lai et al. (2018), Chang et al. (2019) and Cheng et al. (2020).
[b]For a fair comparison, we reproduce the results using their released implementation codes and configurations on the same datasets.

Table 7
Performance of the proposed SDPANet model on Exchange-Rate dataset.

Dataset		Exchange-Rate			
		Horizon			
Methods	Metrics	3	6	12	24
AR[a]	RSE	0.0228	0.0279	0.0353	0.0445
	CORR	0.9734	0.9656	0.9526	0.9357
RNN-GRU[a]	RSE	0.0192	0.0264	0.0408	0.0626
	CORR	0.9786	**0.9712**	0.9531	0.9223
LSTNet-Skip[a]	RSE	0.0226	0.0280	0.0356	0.0449
	CORR	0.9735	0.9658	0.9511	0.9354
LSTNet-Attn[a]	RSE	0.0276	0.0321	0.0448	0.0590
	CORR	0.9717	0.9656	0.9499	0.9339
MTNet[a]	RSE	0.0212	0.0258	0.0347	0.0442
	CORR	0.9767	0.9703	0.9561	0.9388
StemGNN[b]	RSE	0.0687 ± 0.0012	0.0729 ± 0.0016	0.0985 ± 0.0004	0.1139 ± 0.0004
	CORR	0.8535 ± 0.0007	0.8420 ± 0.0003	0.6521 ± 0.0011	0.5571 ± 0.0004
ST-Norm[b]	RSE	0.0710 ± 0.0010	0.0625 ± 0.0005	0.0485 ± 0.0065	0.0585 ± 0.0015
	CORR	0.3210 ± 0.1450	0.1990 ± 0.0010	0.2040 ± 0.0070	0.1675 ± 0.0265
MLCNN[b]	RSE	0.0212 ± 0.0003	0.0280 ± 0.0013	0.0420 ± 0.0006	0.0522 ± 0.0025
	CORR	0.9769 ± 0.0003	0.9674 ± 0.0004	0.9488 ± 0.0008	0.9329 ± 0.0020
SDPANet	RSE	**0.0186 ± 0.0003**	**0.0250 ± 0.0001**	**0.0341 ± 0.0003**	**0.0441 ± 0.0002**
	CORR	**0.9784 ± 0.0004**	0.9706 ± 0.0003	**0.9564 ± 0.0006**	**0.9391 ± 0.0004**

[a]The results are retrieved from Lai et al. (2018), Chang et al. (2019) and Cheng et al. (2020).
[b]For a fair comparison, we reproduce the results using their released implementation codes and configurations on the same datasets.

Fig. 9 illustrates an example of the importance of SR in sales prediction. For an input sales MTS $X_{1\sim t}^{1\sim N}$ with $N = 22$, 5 time series $X_1 \sim X_5$ are selected using SR. Any 2 of the 5 time series pass the Co-Integration test (ex. the P-Value between X_1 and X_5 are smaller than 0.05), which indicates that there are linear correlations between the 5 time series. The series all have similar trend patterns, with sales in the first half period being high, before peaking in the middle period, then remaining relatively low but starting to grow at the end of the time period. Obviously, there is a delay in the linear correlations between the series, which could be modeled by using simultaneous regression. SDPANet-SR removes SR from SDPANet. The weight-sum of the predictions of both SR and SDPANet-SR significantly improve the performances, as indicated by MAE, RSE and CORR, which further confirms the effectiveness of the proposed SR component.

Another case study of comparing SDK of SDPANet and CNN of MLCNN can be seen in Fig. 10. For a MTS with 4 time series X_1, X_2, X_3 and X_4, assume the filter number is set as 4, then each convolutional window (red rectangle) will be assigned 4 specific kernels with size as 3×4. Different colors in each cell of a kernel represents different kernel weights. Compared with MLCNN, kernel weights of SDK are dynamically changing along the times. The top part of Fig. 10(a) and (b) represents the filter output

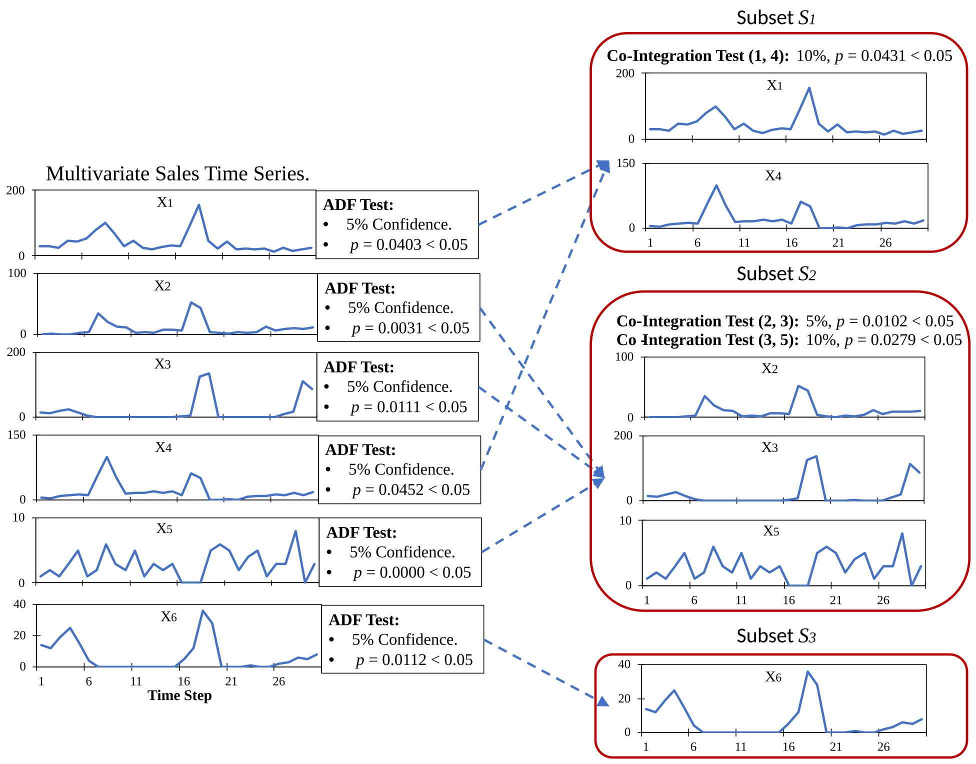


Fig. 8. Case study of SR component.

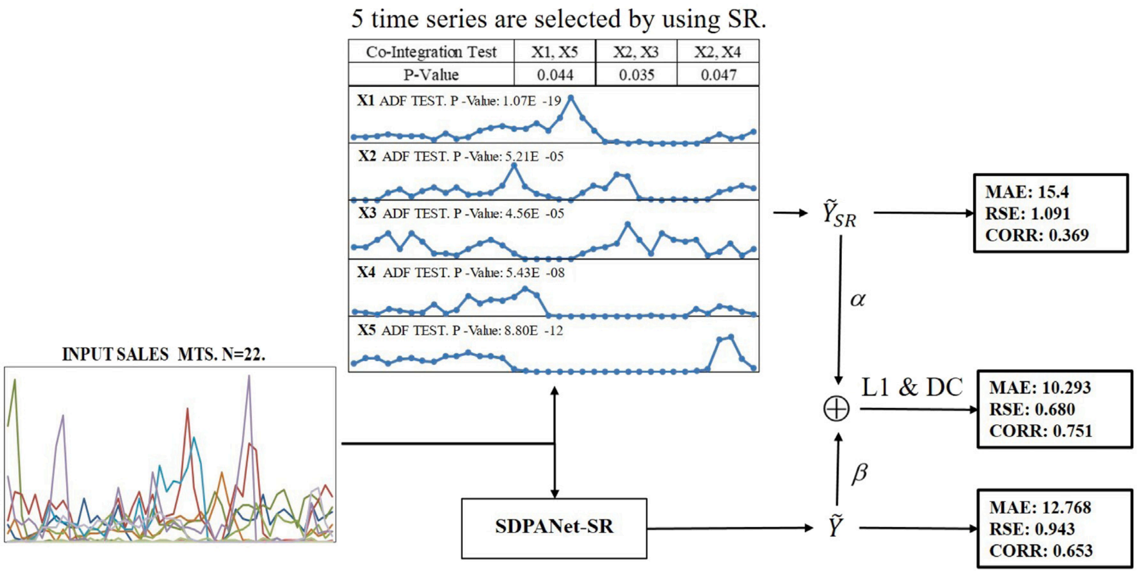

5 time series are selected by using SR.
Co-Integration Test | X1, X5 | X2, X3 | X2, X4
P-Value | 0.044 | 0.035 | 0.047
X1 ADF TEST. P-Value: 1.07E -19
X2 ADF TEST. P-Value: 5.21E -05
X3 ADF TEST. P-Value: 4.56E -05
X4 ADF TEST. P-Value: 5.43E -08
X5 ADF TEST. P-Value: 8.80E -12
INPUT SALES MTS. N=22.
SDPANet-SR
$\tilde{Y}_{SR}$
α
β
$\tilde{Y}$
L1 & DC
MAE: 15.4 RSE: 1.091 CORR: 0.369
MAE: 10.293 RSE: 0.680 CORR: 0.751
MAE: 12.768 RSE: 0.943 CORR: 0.653

Fig. 9. An example showing the important role of SR in sales prediction.

after CNN and SDK convolution operations. Compared with filter output based on dynamic convolution and traditional CNN, it is obvious that the output of SDK has a significant correlation with the changing patterns of original time series.

Fig. 11(a) and (b) show two examples of HA being applied to sales prediction. The attention weight distributions of HA are more closely correlated to the target time series than DARNN. In Fig. 11(a), the level of sales to be predicted is relatively high, and its HA attention is focused on the time point of high historical sales. In Fig. 11(b), the sales to be predicted is relatively small, and its HA attention is focused on the time point of low historical sales. The two examples show that HA component can effectively detect

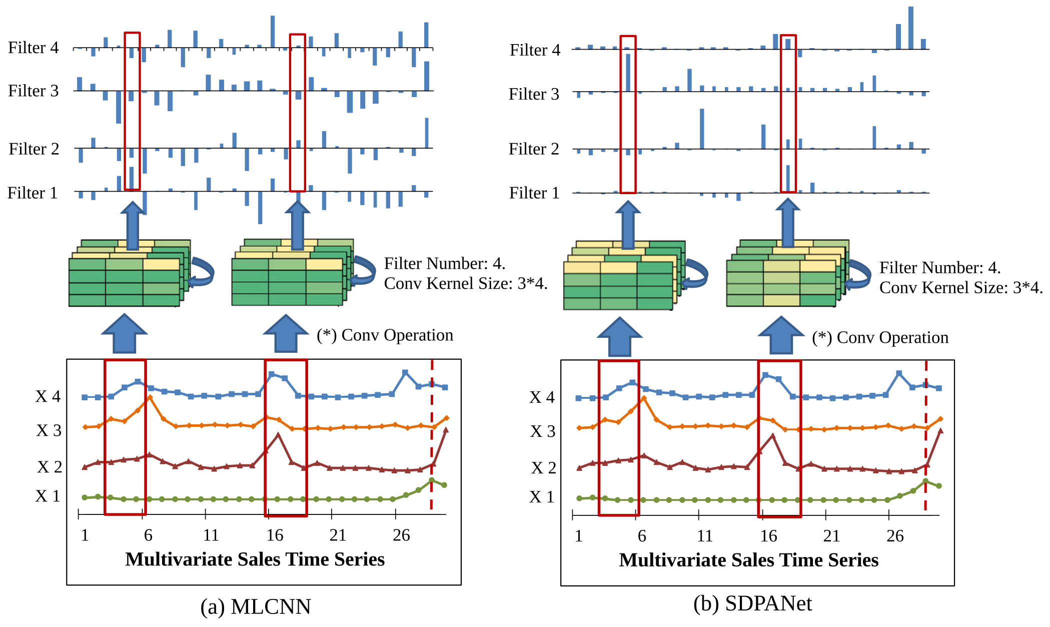


Fig. 10. Case studies of SDK component.

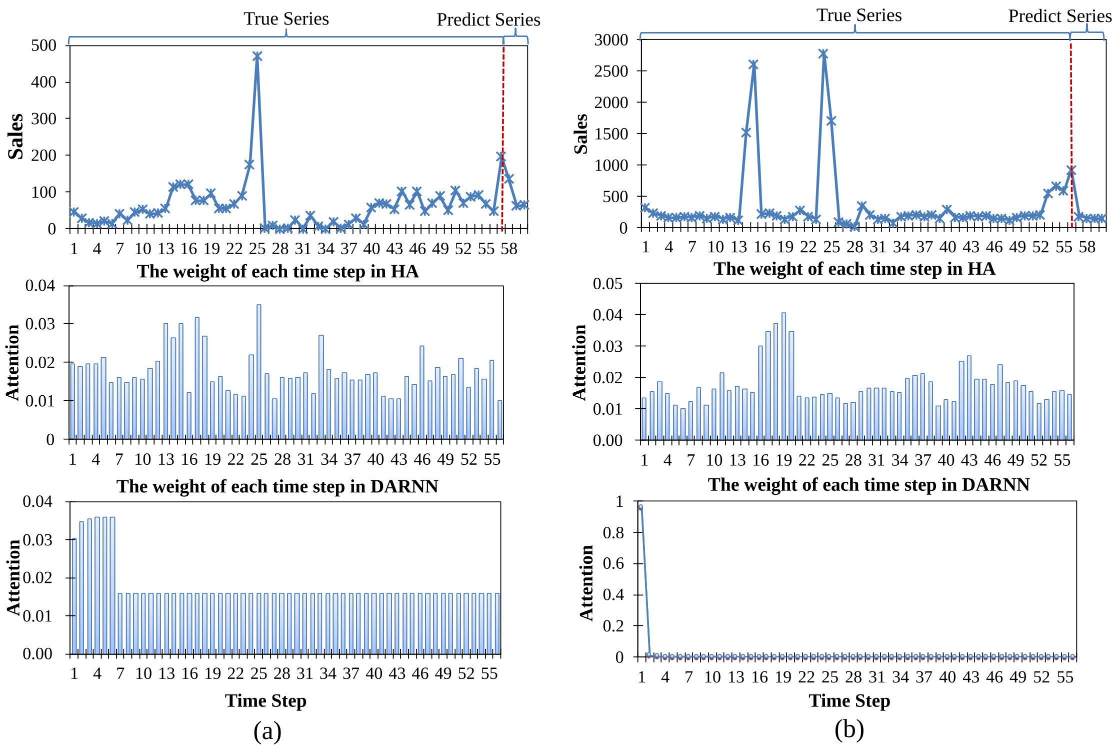


Fig. 11. Pattern analysis based on HA component.

meaningful patterns that are highly correlated with future sales. DARNN however, fails to generate effective attention weights for either target time series. The attention mechanism of HA is similar to that of DARNN: they both contain spatiotemporal attentions. The main difference is that HA uses the output of SDK as its input, while DARNN uses the output of traditional CNN. The examples shown in Fig. 11 provide further evidence that SDK can help to capture more useful dependent correlations.

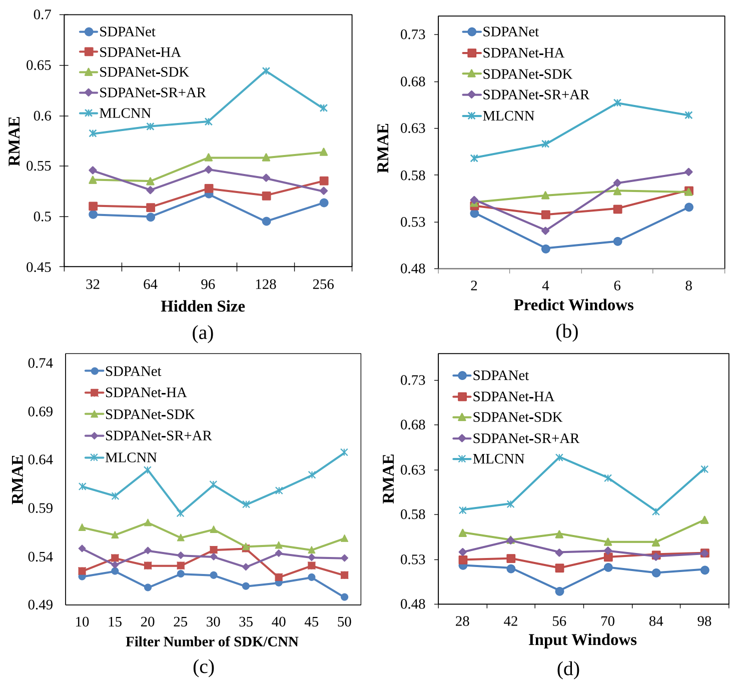


Fig. 12. Results of parameter sensitivity tests.

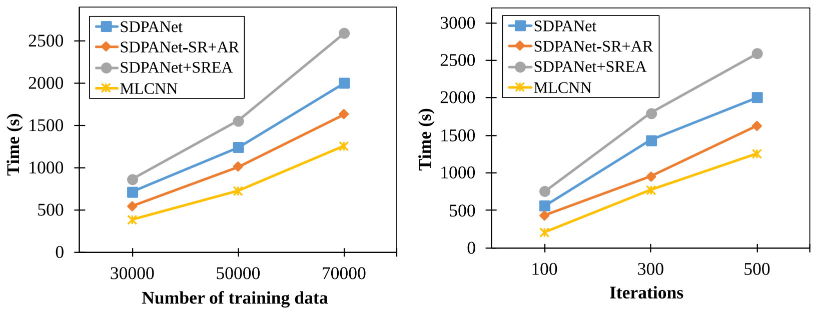


Fig. 13. Running efficiency of SDPANet.

4.7. Parameter sensitive analysis

Parameter sensitive analysis in SPDANet as well as its variants on Galanz dataset could be seen in Fig. 12. The optimal hidden size is set at 128; the optimal filter number is set at 50; the optimal input window is set at 56; the optimal predict time length is set at 4 respectively. Compared with the MLCNN and the variants, our model is less sensitive to the parameter changes, showing the effectiveness of the proposed deep learning framework.

4.8. Running efficiency

Running efficiency of each component is conducted in this section. SDPANet and its 2 variants, SDPANet-SR+AR and SD-PANet+SREA are selected. The new variant SDPANet+SREA adopts a parallel double-loop algorithm to replace parallel dynamic programming of SR component. In addition, MLCNN is also taken as a benchmark. The experiment is executed on a server with 48 cpu cores and 8 GPUs. The number of processors in python multi-processor pool is assigned as 10. All models were iterated 500 times on Cainiao dataset, and we also compare the running efficiency of all models as a function of the number of training data. The experimental results can be seen in Fig. 13.

Compared with SDPANet+SREA, SDPANet adopts parallel dynamic programming strategy, and can improve 20% running efficiency on average. Because the optimized dynamic programming can significantly reduce the number of repeated calculations when adopting multi-processor. In another aspect, the running efficiency of SDPANet is also close to that of SDPANet-SR+AR.

This indicates that adopting parallel computing strategy can significantly reduce the efficiency reduction caused by large-scale co-Integration tests. If the parallel strategy is not adopted, time consuming of model training will be very large (More than tens of thousands of seconds). Compared with MLCNN, SDPANet-SR+AR model also indicates that incorporating SDK, HA, DC components will not increase complexity too much.

5. Discussion

5.1. Theoretical contribution

The theoretical contribution of this paper fall within three main areas. Firstly, our model adds to existing approaches that use convolution network to extract dependent correlations from MTS by extending the application of dynamic kernel from computer vision to time series predictions. We also provide a theoretical analysis using formulas (7) and (8) to explain how, by combining characteristics of sales predictions, we improve on traditional dynamic kernels. The SDK has been shown to be more flexible in fitting changes of time series caused by the influence of a complex external environment, which cannot be properly observed due to uncertain factors. Earlier research considered that MTS could use patterns of commonness, or difference between time series, to reflect the impact of external changes on the predicted value. Inspired by their research, we have illustrated that HA combined with SDK can detect more useful correlation patterns between time series from a larger parameter space, and the larger parameter space indicates that the potential generalization capability of the model is stronger (Cen et al., 2019; Liang et al., 2020).

Secondly, existing deep learning-based MTS prediction models seldom consider stable linear correlations between MTS. One main challenge is that the linear correlations between time series are dynamic, and change over time. Thus, the SR model with its dynamic co-integration mechanism, is proposed here to address the problem.

Finally, traditional DNN-based MTS prediction models use L1 and L2-based loss functions. This often leads to a situation where the model's predictions are very accurate at some time points, while at others there are large deviations from the true values. The reason is that the L1 and L2-based loss functions only considers the deviation between true and predicted values at a specific time point, but seldom consider correlations between different future time points, which are essential to ensure consistency between the predicted and the true series. In this paper, we propose a novel DC loss to make up for the lack of research on this problem.

5.2. Practical implication

The value of this study lies in its excellent performance in predicting sales, giving it the potential to make a significant contribution to enterprises' decision making. The proposed SDPANet model has relevance to the inventory optimization of enterprises because the forecasting of future sales specific to a particular warehouse has the potential to streamline inventory planning, which could reduce inventory costs. The recent five months' online tests show that, compared with manual strategies and with a traditional machine learning based ensemble learning strategy used by some enterprises, use of SDPANet could significantly improve online performances, and could result in large savings of inventory cost. In addition, the proposed model can detect more non-linear and linear patterns that are significantly correlated to future sales trends. Such patterns can improve managers' understanding of the market, helping them make more accurately informed decisions. Finally, the proposed SDK, HA, SR and DC components also have strong practical values in MTS prediction tasks applied in other fields, such as trend predictions of traffic, exchange-rate and etc.

6. Conclusion

In this paper, we proposed a novel SDPANet model for sales prediction based on its unique characteristics. Compared with other MTS, the sales time series is more sensitive to changes from the external environment, and the correlation patterns between time series are more complex. Thus, we have designed a novel Spatiotemporal Dynamic Kernel (SDK) component to extract more useful features from sales MTS to fit the complex non-linear correlations between time series. A novel Hierarchical Attention (HA) component is then proposed to further select important features from SDK through a spatiotemporal attention mechanism for predicting sales. We have also designed a Simultaneous Regression (SR) component to detect dynamic linear stable correlations between time series of MTS. In addition, a DC loss was designed to solve the problem of existing L1 and L2-loss, which are widely applied in current MTS prediction. Experimental results on Galanz and Cainiao verify the effectiveness of SDPANet with average 41.5% reduction on RMAE, average 39.5% reduction on RRSE and average 46% improvement on CORR. Experiments conducted on Traffic and Exchange-Rate further verifies the strong generalization capability of the proposed model.

In the future, we will consider combining stochastic process theory with deep learning to detect more stable non-linear correlations from MTS and verify the new model on more time series from other application domains, we will also attempt to adopt new structures to further accelerate the training speed. In addition, meta-learning, transfer learning and data argument will be taken into considerations. This, we anticipate, will further improve predictions.

CRediT authorship contribution statement

Daifeng Li: Conceptualization, Methodology, Funding acquisition, Software, Resources, Supervision, Writing – review & editing. **Kaixin Lin:** Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing – original draft. **Xuting Li:** Investigation, Resources, Data curation. **Jianbin Liao:** Visualization, Validation. **Ruo Du:** Project administration, Resources. **Dingquan Chen:** Project administration, Supervision. **Andrew Madden:** Writing – review & editing.

Acknowledgments

This article was supported by the National Natural Science Foundation of China (Grant No. 72074231). Soft Science Foundation of Guangdong Province, China (Grant No. 2019A101002020). Key Project of "Pandeng" Program in Guangdong Province, China (Grant No. pdjh2021a0001).

References

Bello-Orgaz, G., M. Mesas, R., Zarco, C., Rodriguez, V., Cordón, O., & Camacho, D. (2020). Marketing analysis of wineries using social collective behavior from users' temporal activity on Twitter. *Information Processing & Management*, *57*(5), Article 102220. http://dx.doi.org/10.1016/j.ipm.2020.102220.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons, http://dx.doi.org/10.5555/3454287.3454722.

Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., et al. (2020). Spectral temporal graph neural network for multivariate time-series forecasting. In *Thirty-third conference on neural information processing systems*. URL https://proceedings.neurips.cc/paper/2020/file/cdf6581cb7aca4b7e19ef136c6e601a5-Paper.pdf.

Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J., & Tang, J. (2019). Representation learning for attributed multiplex heterogeneous network. In *KDD'19* (pp. 1358–1368). URL https://dl.acm.org/doi/pdf/10.1145/3292500.3330964.

Chang, Y.-Y., Sun, F.-Y., Wu, Y.-H., & Shou-De, L. (2019). A memory-network based solution for multivariate time-series forecasting. In *AAAI'19* (pp. 835–844).

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020). Dynamic convolution: Attention over convolution kernels. In *2020 International conference on computer vision and pattern recognition*. http://dx.doi.org/10.1109/CVPR42600.2020.01104.

Chen, L., Liu, Y., Zheng, Z., & Yu, P. (2018). Heterogeneous neural attentive factorization machine for rating prediction. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 833–842). http://dx.doi.org/10.1145/3269206.3271759.

Cheng, J., Huang, K., & Zheng, Z. (2020). Towards better forecasting by fusing near and distant future visions. *34*, In *Proceedings of the AAAI Conference on Artificial Intelligence* (04), (pp. 3593–3600). http://dx.doi.org/10.1609/aaai.v34i04.5766.

Christopher Westland, J., Mou, J., & Yin, D. (2019). Demand cycles and market segmentation in bicycle sharing. *Information Processing & Management*, *56*(4), 1592–1604. http://dx.doi.org/10.1016/j.ipm.2018.09.006.

Cuturi, M., & Blondel, M. (2017). Soft-DTW: A differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning*.

Das, M., & Ghosh, S. K. (2017). Sembnet: A semantic Bayesian network for multivariate prediction of meteorological time series data. *Pattern Recognition Letters*, *93*, 192–201. http://dx.doi.org/10.1016/j.patrec.2017.01.002.

Dasgupta, S., & Osogami, T. (2017). Nonlinear dynamic Boltzmann machines for time-series prediction. *31*, In *Proceedings of the AAAI conference on artificial intelligence*. (1), URL https://ojs.aaai.org/index.php/AAAI/article/view/10806.

Deng, J., Chen, X., Jiang, R., Song, X., & Tsang, I. W. (2021). ST-norm: Spatial and temporal normalization for multi-variate time series forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 269–278). URL https://dl.acm.org/doi/abs/10.1145/3447548.3467330.

Ekambaram, V., Manglik, K., Mukherjee, S., Sajja, S. S. K., Dwivedi, S., & Raykar, V. (2020). Attention based multi-modal new product sales time-series forecasting. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 3110–3118). http://dx.doi.org/10.1145/3394486.3403362.

Fan, C., Zhang, Y., Pan, Y., Li, X., Zhang, C., Yuan, R., et al. (2019). Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2527–2535). http://dx.doi.org/10.1145/3292500.3330662.

Gao, H., Kong, D., Lu, M., Bai, X., & Yang, J. (2018). Attention convolutional neural network for advertiser-level click-through rate forecasting. In *Proceedings of the 2018 world wide web conference* (pp. 1855–1864). http://dx.doi.org/10.1145/3178876.3186184.

He, Y., Zhu, C., Wang, M., Marios, S., & Zhang, X. (2019). Bounding box regression with uncertainty for accurate object detection. In *CVPR'19, Conference on computer vision and pattern recognition 2019*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2020). Squeeze-and-excitation networks. *42*, In *Proceedings of the IEEE conference on computer vision and pattern recognition* (8), (pp. 2011–2023). http://dx.doi.org/10.1109/TPAMI.2019.2913372.

Hu, W., Yang, Y., Wang, J., Huang, X., & Cheng, Z. (2020). Understanding electricity-theft behavior via multi-source data. In *Proceedings of the world wide web conference (WWW'20). 20-24 April*. http://dx.doi.org/10.1145/3366423.3380291.

Huang, S., Wang, D., Wu, X., & Tang, A. (2019). DSANet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 2129–2132). http://dx.doi.org/10.1145/3357384.3358132.

Kaya, K., Yılmaz, Y., Yaslan, Y., Öğüdücü, Ş. G., & Çıngı, F. (2021). Demand forecasting model using hotel clustering findings for hospitality industry. *Information Processing & Management*, *59*, http://dx.doi.org/10.1016/j.ipm.2021.102816.

Kendall, A., Gal, Y., & Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7482–7491). http://dx.doi.org/10.1109/CVPR.2018.00781.

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (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* (pp. 95–104). http://dx.doi.org/10.1145/3209978.3210006.

Laptev, N., Yosinski, J., Li, L. E., & Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning, Vol. 34* (pp. 1–5). URL http://roseyu.com/time-series-workshop/submissions/TSW2017_paper_3.pdf.

Lee, H., Jin, S., Chu, H., Lim, H., & Ko, S. (2021). Learning to remember patterns: Pattern matching memory networks for traffic forecasting. In *Tenth International Conference on Learning Representations*. URL https://arxiv.org/pdf/2110.10380.pdf.

Li, X., Wu, P., & Wang, W. (2020). Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. *Information Processing & Management*, *57*(5), Article 102212. http://dx.doi.org/10.1016/j.ipm.2020.102212.

Li, H., Xu, Z., Taylor, G., Studer, C., & Tom, G. (2018). Visualizing the loss landscape of neural nets. In *NIPs'18, Proceedings of the 32nd conference on neural information processing systems* (pp. 6391–6401).

Liang, X., Li, D., & Madden, A. (2020). Attributed network embedding based on mutual information estimation. In *CIKM'20* (pp. 835–844). URL https://dl.acm.org/doi/pdf/10.1145/3292500.3330964.

Liang, Z., Mao, J., Lu, K., Bai, Z., & Gang, L. (2021). Combining deep neural network and bibliometric indicator for emerging research topic prediction. *Information Processing & Management*, *58*(5), 1–18, URL https://www.sciencedirect.com/science/article/pii/S0306457321001072#!.

Lim, B., Ark, S., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, (1), http://dx.doi.org/10.1016/j.ijforecast.2021.03.012.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., et al. (2021). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*. URL https://openreview.net/forum?id=0EXmFzUn5I.

Mellit, A., Pavan, A. M., & Benghanem, M. (2013). Least squares support vector machine for short-term prediction of meteorological time series. *Theoretical and Applied Climatology*, *111*(1), 297–307, URL https://linkspringer.fenshishang.com/article/10.1007/s00704-012-0661-7.

Nguyen, L., Pan, Z., Openiyi, O., Abu-gellban, A., Moghadasi, M., & Jin, F. (2020). Self-boosted time-series forecasting with multi-task and multi-view learning. In *Proceedings of the thirty-fourth AAAI conference on artificial intelligence. AAAI'20. 7-12 February. New York. USA*. URL https://arxiv.org/pdf/1909.08181.pdf.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Eighth international conference on learning representations*. URL https://arxiv.org/pdf/1905.10437.pdf.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. In *Twenty-sixth international joint conference on artificial intelligence*. URL https://arxiv.org/pdf/1704.02971.pdf.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., & Aigrain, S. (2019). Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society, Series A*, *371*, http://dx.doi.org/10.1098/rsta.2011.0550.

Sakoe, H., & ChibaLim, S. (1990). Dynamic programming algorithm optimization for spoken word recognition. *Read. Speech Recognit.*, 159–224. http://dx.doi.org/10.1109/TASSP.1978.1163055.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191. http://dx.doi.org/10.1016/j.ijforecast.2019.07.001.

Shen, Z., Yuan, R., Wu, D., & Pei, J. (2018). Data science in retail-as-a-service. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (KDD'18 tutorial speak). London. UK*. http://dx.doi.org/10.1145/3219819.3219943.

Shi, J., Yao, H., Wu, X., Li, T., Lin, Z., Wang, T., et al. (2021). Relation-aware meta-learning for E-commerce market segment demand prediction with limited records. In *The fourth ACM international conference on web search and data mining*. http://dx.doi.org/10.1145/3437963.3441750.

Shi, Q., Yin, J., Cai, J., Cichocki, A., Yokota, T., Chen, L., et al. (2020a). Block Hankel tensor ARIMA for multiple short time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 34* (04), (pp. 5758–5766). http://dx.doi.org/10.1609/aaai.v34i04.6032.

Shi, Q., Yin, J., Cai, J., Cichocki, A., Yokota, T., Chen, L., et al. (2020b). Block Hankel tensor ARIMA for multiple short time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 34* (04), (pp. 5758–5766). http://dx.doi.org/10.1609/aaai.v34i04.6032.

Song, C., Lin, Y., Guo, S., & Wan, H. (2020). Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the 34th AAAI conference on artificial intelligence (AAAI'20)*. URL https://ojs.aaai.org/index.php/AAAI/article/download/5438/5294.

Tang, X., Yao, H., Sun, Y., Aggarwal, C., Mitra, P., & Wang, S. (2020). Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 34* (04), (pp. 5956–5963). http://dx.doi.org/10.1609/aaai.v34i04.6056.

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45. http://dx.doi.org/10.1080/00031305.2017.1380080.

Vallance, L., Charbonnier, B., Paul, N., Dubost, S., & Blanc, P. (2017). Towards a standardized procedure to assess solar forecast accuracy: A new ramp and time alignment metric. *Solar Energy*, *150*, 408–422. http://dx.doi.org/10.1016/j.solener.2017.04.064.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *NIPS'17*, *Proceedings of the 31st international conference on neural information processing systems* (pp. 6000–6010). Red Hook, NY, USA: Curran Associates Inc., URL https://dl.acm.org/doi/10.5555/3295222.3295349.

Yakhchi, S., Behehsti, A., mohssen Ghafari, S., Razzak, I., Orgun, M., & Elahi, M. (2022). A convolutional attention network for unifying general and sequential recommenders. *Information Processing & Management*, *59*(1), URL https://www.sciencedirect.com/science/article/pii/S0306457321002363.

Ye, M., Luo, J., Xiao, C., & Ma, F. (2020). LSAN: Modeling long-term dependencies and short-term correlations with hierarchical attention for risk prediction. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1753–1762). http://dx.doi.org/10.1145/3340531.3411864.

Yu, J., Jiang, Y., Wang, Z., Cao, Z., & Huang, T. (2016). Unitbox: An advanced object detection network. In *MM'16*, *Proceedings of the 24th ACM international conference on multimedia* (pp. 516–520). New York, NY, USA: http://dx.doi.org/10.1145/2964284.2967274.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021a). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the thirty-fifth AAAI conference on artificial intelligence*. URL https://www.aaai.org/AAAI21Papers/AAAI-7346.ZhouHaoyi.pdf.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021b). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*. URL https://www.aaai.org/AAAI21Papers/AAAI-7346.ZhouHaoyi.pdf.