The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Predicting Temporal Sets with Simplified Fully Connected Networks

Le Yu, Zihang Liu, Tongyu Zhu*, Leilei Sun, Bowen Du, Weifeng Lv

State Key Laboratory of Software Development Environment, Beihang University, Beijing, 100191, China
{yule,lzhmark,zhutongyu,leileisun,dubowen,lwf}@buaa.edu.cn

Abstract

Given a sequence of sets, where each set contains an arbitrary number of elements, temporal sets prediction aims to predict which elements will appear in the subsequent set. Existing methods for temporal sets prediction are developed on sophisticated components (e.g., recurrent neural networks, attention or gating mechanisms, and graph neural networks), which inevitably increase the model complexity due to more trainable parameters and higher computational costs. Moreover, the involved nonlinear activation may contribute little or even degrade the performance. In this paper, we present a succinct architecture that is solely built on the Simplified Fully Connected Networks (SFCNs) for temporal sets prediction to bring both effectiveness and efficiency together. In particular, given a user's sequence of sets, we employ SFCNs to derive representations of the user by learning inter-set temporal dependencies, intra-set element relationships, and intra-embedding channel correlations. Two families of general functions are introduced to preserve the permutation-invariant property of each set and the permutation-equivariant property of elements in each set. Moreover, we design a user representations adaptive fusing module to aggregate user representations according to each element for improving the prediction performance. Experiments on four benchmarks show the superiority of our approach over the state-of-the-art under both transductive and inductive settings. We also theoretically and empirically demonstrate that our model has lower space and time complexity than baselines. Codes and datasets are available at https://github.com/yule-BUAA/SFCNTSP.

Introduction

Temporal sets can be defined as a sequence of sets, where each set has a timestamp and includes an arbitrary number of elements (Benson, Kumar, and Tomkins 2018). In practice, when customers purchase baskets of products (Rendle, Freudenthaler, and Schmidt-Thieme 2010; Yu et al. 2016), patients take medical prescriptions (Baytas et al. 2017; Jin et al. 2018), students select courses (Lin et al. 2017) or travelers choose attractions (Zhu et al. 2021) based on the historical behaviors, they all deal with temporal sets. Making accurate predictions of which elements will appear in the next-period set could help people make better decisions (Hu and He 2019; Sun et al. 2020; Yu et al. 2020, 2022).

In recent years, a number of efforts have been made on temporal sets prediction. A part of the approaches followed a two-step strategy by first obtaining the representations of sets via pooling operations or matrix factorization, and then learning the temporal dependencies in user behaviors based on Recurrent Neural Networks (RNNs) or the attention mechanism (Yu et al. 2016; Hu and He 2019; Shi et al. 2021). Another part of the methods additionally learned on elements. Sun et al. (2020) presented a Transformer-based dual sequential network to learn the element-level and set-level representations for each user. Yu et al. (2020) first constructed a sequence of element snapshots according to their co-occurrence and then learned on the snapshots by Graph Neural Networks (GNNs), the attention mechanism and the gating mechanism. Yu et al. (2022) connected the sequences of different users by a temporal graph and devised an element-guided message aggregation mechanism with the usage of temporal information.

Existing methods for temporal sets prediction are usually designed with complicated modules, such as RNNs, the attention or gating mechanism, and GNNs. Although these methods have achieved remarkable performance, they inevitably introduce more trainable parameters and incur higher computational costs. As shown in Table 1, the theoretical space and time complexity of the existing methods are expensive in most cases. Moreover, in the temporal sets prediction problem, each element is only associated with an identifier (ID) without given semantic features. Therefore, the nonlinear activation function involved in previous models might become helpless or even harmful to the performance, which has been observed in the fields of graph learning (Wu et al. 2019) and recommender systems (He et al. 2020). The above phenomena motivate us to consider *whether it is necessary to design sophisticated components with the nonlinear activation for temporal sets prediction.*

In this paper, we propose a succinct architecture that is solely built on **S**implified **F**ully **C**onnected **N**etworks for **T**emporal **S**ets **P**rediction (SFCNTSP) to bring both effectiveness and efficiency together[1]. Our approach first em-

*Corresponding Author.

[1]We name a single linear fully connected layer as the Simplified Fully Connected Network (SFCN) in this paper.

Methods	Embeddings	Temporal Dependencies		Element Relationships		Channel Correlations	
	Space	Space	Time	Space	Time	Space	Time
Sets2Sets	$\mathcal{O}(nc)$	$\mathcal{O}(c^2L_T)$	$\mathcal{O}(\bar{t}c(\bar{n}+cL_T)$	—	—	—	—
DSNTSP	$\mathcal{O}(nc)$	$\mathcal{O}(c^2L_T)$	$\mathcal{O}(\bar{t}\bar{n}c(c+\bar{t}\bar{n})L_T)$	$\mathcal{O}(c^2L_E)$	$\mathcal{O}(\bar{t}\bar{n}c^2L_E)$	—	—
DNNTSP	$\mathcal{O}(nc)$	$\mathcal{O}(c^2L_T)$	$\mathcal{O}(\bar{t}\bar{n}c(c+\bar{t}\bar{n})L_T)$	$\mathcal{O}(c^2L_E)$	$\mathcal{O}(\bar{t}\bar{n}c(c+\bar{t}+\bar{n})L_E)$	—	—
ETGNN	$\mathcal{O}(mc+nc)$	$\mathcal{O}(c^2L_T)$	$\mathcal{O}(\bar{t}\bar{n}c^2L_T)$	—	$\mathcal{O}(n\bar{t}\bar{n}cL_E)$	—	—
SFCNTSP	$\mathcal{O}(nc)$	$\mathcal{O}(\bar{t}^2)$	$\mathcal{O}(\bar{t}c(\bar{n}+\bar{t}))$	—	$\mathcal{O}(\bar{t}\bar{n}^2c)$	$\mathcal{O}(c^2)$	$\mathcal{O}(\bar{t}\bar{n}c^2)$

Table 1: Space complexity and time complexity of the existing models and our SFCNTSP in each process. L_T and L_E are the numbers of layers in learning temporal dependencies and element relationships. $\bar{t}$, $\bar{n}$, and c denote the average sequence length, the average number of elements in each set, and the number of embedding channels. m and n represent the number of users and elements. Note that in ETGNN, L_T is equal to L_E.

ploys the SFCNs to learn on the sequence of sets of a given user for deriving his/her representations and then computes the appearing probabilities of all the elements in the next-period set by adaptively fusing the representations. The permutation-invariant property of each set and permutation-equivariant property of each element within the set (Zaheer et al. 2017; Lee et al. 2019) are also well preserved by our method. To be specific, we first present a family of permutation-invariant functions to obtain the vectorized representation of each set and then utilize an SFCN to capture the inter-set temporal dependencies in the sequence. Next, we define a family of permutation-equivariant functions to enable elements within the same set to interact with each other for learning intra-set element relationships. Then, we leverage another SFCN to exploit the implicit correlations of different embedding channels. Finally, we adaptively aggregate representations of the user according to each element to improve the performance. Extensive experiments on real-world datasets show that our approach could significantly outperform existing methods under both transductive and inductive settings with fewer trainable parameters and lower computational costs. Overall, our key contributions include:

- We show that the temporal sets prediction problem can be well addressed by a succinct architecture, which is solely based on the SFCNs and could learn the inter-set temporal dependencies, intra-set element relationships, and intra-embedding channel correlations.
- We present two families of general functions to guarantee the permutation-invariant property of each set and the permutation-equivariant property of elements in each set.
- We design a user representations adaptive fusing module to aggregate representations of the user with respect to each element for facilitating the prediction task.

Preliminaries

Let $\mathbb{U} = \{u_1, \cdots, u_m\}$ and $\mathbb{V} = \{v_1, \cdots, v_n\}$ be the collections of m users and n elements, respectively. A set $\mathbb{S} \subset \mathbb{V}$ is a collection of elements. Let $\mathbb{P}_\mathbb{S}$ denote the collection of all the $|\mathbb{S}|!$ permutations of indices $\{1, 2, \cdots, |\mathbb{S}|\}$.

Permutation-invariance. For a set $\mathbb{S}$, the representation of $\mathbb{S}$ remains the same, regardless of the order of elements in the set. Formally, for any permutation $\pi \in \mathbb{P}_\mathbb{S}$, the function $f(\cdot)$ should satisfy $f(\mathbb{S}) = f(\pi\mathbb{S})$, where $f(\cdot)$ aims to embed $\mathbb{S}$ into a fixed-length vector.

Permutation-equivariance. For a set $\mathbb{S}$, the representation of each element in $\mathbb{S}$ keeps the same, regardless of the order of elements in the set. Formally, for any permutation $\pi \in \mathbb{P}_\mathbb{S}$, the function $g(\cdot)$ should satisfy $g(\pi\mathbb{S}) = \pi g(\mathbb{S})$, where $g(\cdot)$ learns the representation for each element in $\mathbb{S}$.

Temporal Sets Prediction. Given a sequence of sets $S = \{\mathbb{S}^1, \mathbb{S}^2, \cdots, \mathbb{S}^t\}$ that records the historical behaviors of user $u \in \mathbb{U}$, temporal sets prediction aims to design a model to predict which elements will appear in the subsequent set $\hat{\mathbb{S}}^{t+1}$ according to the historical sequence S. The model should preserve the permutation-invariant property of each set and the permutation-equivariant property of each element in the set. Note that in this paper, we use the relative time index for numbering user records.

Methodology

The framework of the proposed model is shown in Figure 1, which is composed of four components: inter-set temporal dependencies learning, intra-set element relationships learning, intra-embedding channel correlations learning, and user representations adaptive fusing. In particular, given a user's sequence of sets, we first obtain the representations of elements in the sets via the element embedding layer. Then, we present a family of permutation-invariant functions to derive the representation of each set and employ an SFCN to capture the temporal dependencies across different sets. Next, we introduce a family of permutation-equivariant functions to learn the relationships of elements within each set by allowing elements to interact with each other. Then, we utilize another SFCN to exploit the implicit correlations of multiple embedding channels. Finally, based on the user representations learned from the above modules, we adaptively aggregate representations of the user according to each element and compute its appearing probability in the next-period set.

SFCN As the Backbone

We first illustrate why we use SFCN rather than the Fully Connected Network (FCN) or Multi-Layer Perceptron (MLP) as the backbone. Firstly, both FCN and MLP are multi-layered with a series of fully connected layers. Compared with FCN, MLP additionally involves the nonlinear activation function between every two adjacent layers, which might be unnecessary for predicting temporal sets (explained in the Introduction). Therefore, MLP is not cho-

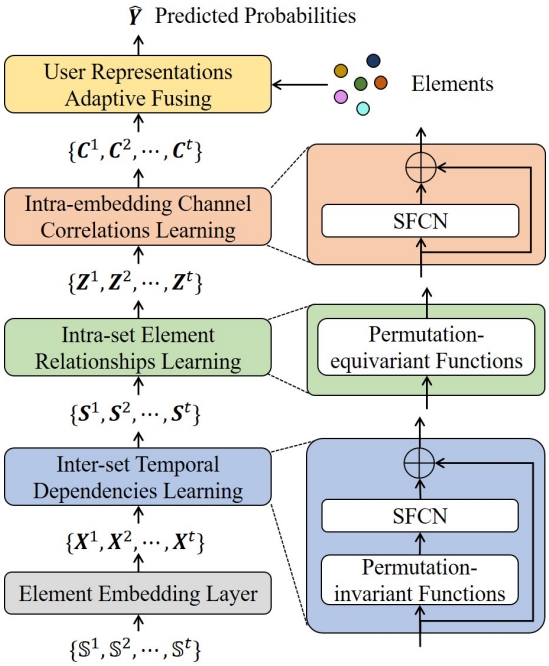

Figure 1: Framework of the proposed model.

sen. Secondly, for an input $\boldsymbol{X}$, the computation of an L-layered FCN can be denoted by $\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W}^{(1)} \cdots \boldsymbol{W}^{(L)}$ with $\boldsymbol{W}^{(l)}$ as trainable weights, $1 \leq l \leq L$. As linearly multiplying multiple trainable weights is theoretically equal to multiplying a single trainable weight, we abandon the multi-layered FCN and build our framework by the single-layered linear SFCN with trainable weight $\boldsymbol{W} = \boldsymbol{W}^{(1)} \cdots \boldsymbol{W}^{(L)}$. Our assumptions will be validated in the experiments.

Inter-set Temporal Dependencies Learning

Let $\boldsymbol{E} \in \mathbb{R}^{n \times c}$ denote the embedding lookup table of all the n elements with c as the number of embedding channels. Given the sequence of sets $\mathcal{S} = \left\{\mathbb{S}^1, \mathbb{S}^2, \cdots, \mathbb{S}^t\right\}$ of user $u \in \mathbb{U}$, we retrieve the embeddings of elements in each set from $\boldsymbol{E}$ and obtain the input sequence $\left\{\boldsymbol{X}^1, \boldsymbol{X}^2, \cdots, \boldsymbol{X}^t\right\}$ with $\boldsymbol{X}^{t'} \in \mathbb{R}^{n^{t'} \times c}$, where $1 \leq t' \leq t$ and $n^{t'}$ denotes the number of elements in set $\mathbb{S}^{t'}$.

As each set contains an arbitrary number of elements, we need to first embed each set into a fixed-length vector and then learn on the sequence of vectorized representations. The set representation should keep identical to any order of elements in the set. To this end, we design a family of permutation-invariant functions to compute on each set, which should satisfy the following format,

$$f\left(\widetilde{\boldsymbol{X}}\right) = \mathbf{1}_{\widetilde{n}}^{\top}\left(\boldsymbol{F} \odot \widetilde{\boldsymbol{X}}\right) \in \mathbb{R}^{c}, \tag{1}$$

where $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{\widetilde{n} \times c}$ is the input. $\mathbf{1}_{\widetilde{n}} \in \mathbb{R}^{\widetilde{n}}$ is a $\widetilde{n}$-dimensional all-ones vector. $\boldsymbol{F} \in \mathbb{R}^{\widetilde{n} \times c}$ reflects the contributions of elements in $\widetilde{\boldsymbol{X}}$, which *should not be affected by the order of elements*. We give several instances of $\boldsymbol{F}$: for $\forall 1 \leq i \leq \widetilde{n}, 1 \leq j \leq \widetilde{n}$, 1) $F_{i,j} = 1/\widetilde{n}$ (average pooling); 2) $F_{i,j} = 1$ (sum pooling); 3) $F_{i,j} = 1$ if i equals to $argmax(\widetilde{\boldsymbol{X}}_{*,j})$, 0 otherwise (max pooling); 4) $F_{i,j} = 1$ if i equals to $argmin(\widetilde{\boldsymbol{X}}_{*,j})$, 0 otherwise (min pooling). We can also incorporate a trainable weight $\boldsymbol{q} \in \mathbb{R}^{c}$ to derive $\boldsymbol{F}$ by $\widetilde{\boldsymbol{X}}\boldsymbol{q}\mathbf{1}_{c}^{\top}$, which represents the attention-based pooling. Based on $f(\cdot)$, we obtain the representations of sets $\left\{\overline{\boldsymbol{x}}^1, \overline{\boldsymbol{x}}^2, \cdots, \overline{\boldsymbol{x}}^t\right\}$ by

$$\overline{\boldsymbol{x}}^{t'} = f\left(\boldsymbol{X}^{t'}\right) \in \mathbb{R}^{c}, 1 \leq t' \leq t. \tag{2}$$

In this paper, we simply use the average pooling to implement $f(\cdot)$ (i.e., for $\forall 1 \leq i \leq \widetilde{n}, 1 \leq j \leq \widetilde{n}, F_{i,j} = 1/\widetilde{n}$) and leave the explorations of other instances for future work.

Then, we employ an SFCN to learn the temporal dependencies of sets in the sequence since SFCN is naturally sensitive to the input order and suitable for dealing with sequential data. To be specific, we pack $\left\{\overline{\boldsymbol{x}}^1, \overline{\boldsymbol{x}}^2, \cdots, \overline{\boldsymbol{x}}^t\right\}$ up and denote it as $\overline{\boldsymbol{X}} \in \mathbb{R}^{t \times c}$. The SFCN calculates on $\overline{\boldsymbol{X}}$ by

$$\overline{\overline{\boldsymbol{X}}} = \boldsymbol{W}_T\overline{\boldsymbol{X}} \in \mathbb{R}^{t \times c}, \tag{3}$$

where $\boldsymbol{W}_T \in \mathbb{R}^{t \times t}$ is the trainable weight for learning the importance of each time[2]. Then, we unpack $\overline{\overline{\boldsymbol{X}}}$ to the sequence $\left\{\overline{\overline{\boldsymbol{x}}}^1, \overline{\overline{\boldsymbol{x}}}^2, \cdots, \overline{\overline{\boldsymbol{x}}}^t\right\}$ and adopt the residual connection (He et al. 2016) to incorporate the inputs as follows,

$$\boldsymbol{S}^{t'} = \boldsymbol{X}^{t'} + \mathbf{1}_{n^{t'}}\overline{\overline{\boldsymbol{x}}}^{t'\top} \in \mathbb{R}^{n^{t'} \times c}. \tag{4}$$

Finally, we obtain the output sequence $\left\{\boldsymbol{S}^1, \boldsymbol{S}^2, \cdots, \boldsymbol{S}^t\right\}$ that captures the temporal dependencies of different sets.

Intra-set Element Relationships Learning

Intuitively, elements within the same set are associated with each other and it is essential to learn the intra-set element relationships (Yu et al. 2020). However, it is infeasible to directly employ an SFCN to capture the element relationships because of its sensitivity to the order of elements. Therefore, we present a family of permutation-equivariant functions to compute order-agnostic element representations and make intra-set elements interact with each other, which should have the following format,

$$g\left(\widetilde{\boldsymbol{X}}\right) = \boldsymbol{G}\widetilde{\boldsymbol{X}} \in \mathbb{R}^{\widetilde{n} \times c}, \tag{5}$$

where $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{\widetilde{n} \times c}$ is the input. $\boldsymbol{G} \in \mathbb{R}^{\widetilde{n} \times \widetilde{n}}$ captures the relationships of elements in $\widetilde{\boldsymbol{X}}$, which *should be derived by an order-agnostic manner*. We introduce the following instances of $\boldsymbol{G}$.

$$\boldsymbol{G} = \alpha \cdot diag(\mathbf{1}_{\widetilde{n}}^{\top}) + \beta \cdot (\mathbf{1}_{\widetilde{n}}\mathbf{1}_{\widetilde{n}}^{\top}), \tag{6}$$

where $diag(\cdot)$ returns the diagonal matrix of the input vector. $\alpha, \beta \in \mathbb{R}$ can be predefined hyperparameters or trainable weights. $\boldsymbol{G}$ could be further dependent on the inputs as:

$$G_{i,j} = s(\widetilde{\boldsymbol{X}}_{i,*}, \widetilde{\boldsymbol{X}}_{j,*}), \tag{7}$$

[2]Since different users usually have varied sequence lengths and SFCN can only deal with inputs with a fixed length, when implementing this part, we first count the maximal user sequence length t_{max} and then set the shape of $\boldsymbol{W}_T$ to be $t_{max} \times t_{max}$ as well as padding the length of the input sequence to be t_{max}.

where $s(\boldsymbol{x}, \boldsymbol{y})$ computes the order-agnostic similarities between inputs $\boldsymbol{x}$ and $\boldsymbol{y}$, such as the dot product $s(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}\boldsymbol{y}^\top$ or the cosine similarity $s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}\boldsymbol{y}^\top}{\|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{y}\|_2}$. Given the aforementioned sequence $\left\{\boldsymbol{S}^1, \boldsymbol{S}^2, \cdots, \boldsymbol{S}^t\right\}$, we get the output sequence $\left\{\boldsymbol{Z}^1, \boldsymbol{Z}^2, \cdots, \boldsymbol{Z}^t\right\}$ based on $g(\cdot)$ by

$$\boldsymbol{Z}^{t'} = g(\boldsymbol{S}^{t'}) \in \mathbb{R}^{n^{t'} \times c}. \tag{8}$$

In this paper, we choose the predefined hyperparameters α and β to learn element relationships and leave the investigations of other instances for future work.

Intra-embedding Channel Correlations Learning

We further employ another SFCN to exploit the implicit correlations of multiple channels for the element embedding. Specifically, given the sequence $\left\{\boldsymbol{Z}^1, \boldsymbol{Z}^2, \cdots, \boldsymbol{Z}^t\right\}$, the SFCN computes on $\boldsymbol{Z}^{t'}$ for $1 \leq t' \leq t$ as

$$\overline{\boldsymbol{C}}^{t'} = \boldsymbol{Z}^{t'}\boldsymbol{W}_C \in \mathbb{R}^{n^{t'} \times c}, \tag{9}$$

where $\boldsymbol{W}_C \in \mathbb{R}^{c \times c}$ is the trainable weight for learning the correlations of different channels. We also utilize the residual connection to consider the inputs by

$$\boldsymbol{C}^{t'} = \boldsymbol{Z}^{t'} + \overline{\boldsymbol{C}}^{t'} \in \mathbb{R}^{n^{t'} \times c}. \tag{10}$$

The output sequence is denoted as $\left\{\boldsymbol{C}^1, \boldsymbol{C}^2, \cdots, \boldsymbol{C}^t\right\}$.

User Representations Adaptive Fusing

Let $\mathcal{V} = unique(\mathcal{S}) \subset \mathbb{V}$ denote the collection of elements that have appeared in user u's sequence $\mathcal{S}$. We aim to obtain the representation of element $\widetilde{v} \in \mathcal{V}$ based on $\left\{\boldsymbol{C}^1, \boldsymbol{C}^2, \cdots, \boldsymbol{C}^t\right\}$. Concretely, we first retrieve $\widetilde{v}$'s corresponding representation in $\boldsymbol{C}^{t'}$ if $\widetilde{v}$ appears in the set $\mathbb{S}^{t'}$ with $1 \leq t' \leq t$, and then average the retrieved representations to get $\widetilde{v}$'s final representation $\boldsymbol{h}_{\widetilde{v}} \in \mathbb{R}^c$ by

$$\boldsymbol{h}_{\widetilde{v}} = average(\{retrieve(\boldsymbol{C}^{t'}, \widetilde{v}), \forall 1 \leq t' \leq t\}), \tag{11}$$

where $retrieve(\boldsymbol{C}^{t'}, \widetilde{v})$ returns the corresponding representation of $\widetilde{v}$ if it is contained in the set $\mathbb{S}^{t'}$. Finally, we could obtain the representations of all the elements in $\mathcal{V}$ and pack them up as the user representations $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}| \times c}$. To compute the appearing probability of each element $v \in \mathbb{V}$ in the next-period set, we first adaptively aggregate user representations $\boldsymbol{H}$ based on element v and then compute the similarity of the fused representation with v's embedding $\boldsymbol{e}_v$ (retrieved from the element embedding lookup table $\boldsymbol{E}$) by

$$\hat{Y}_{u,v} = sigmoid(softmax(\sigma(\boldsymbol{H}\boldsymbol{e}_v))^\top \boldsymbol{H}\boldsymbol{W}_P\boldsymbol{e}_v), \tag{12}$$

where $\hat{Y}_{u,v} \in \mathbb{R}$ is the predicted probability that element v will appear in user u's next-period set. $\sigma(\cdot)$ denotes the $LeakyReLU$ function. $\boldsymbol{W}_P \in \mathbb{R}^{c \times c}$ is the trainable projection weight. The adaptive fusing module aggregates user representations for each element, which is more expressive than trivially averaging or adding up the representations.

Model Training Process

We train our model in a mini-batch manner by the padding technique. To be specific, we use t_{max} to denote the maximal user sequence length. Given a batched data $\mathcal{B} = \{\mathcal{S}_i, \forall 1 \leq i \leq |\mathcal{B}|\}$, we use $n_{\mathcal{B}}$ to represent the maximal number of elements in the set in $\mathcal{B}$. Then, we pad for shorter sequences and smaller sets to align their dimensions. Finally, we can obtain a four-dimensional tensor $\boldsymbol{B} \in \mathbb{R}^{|\mathcal{B}| \times t_{max} \times n_{\mathcal{B}} \times c}$ as the model input for mini-batch training.

We represent each element as a label and treat temporal sets prediction as a multi-label classification problem. The ground truth of user u's next-period set is denoted by $\boldsymbol{Y}_{u,*} \in \{0, 1\}^n$, where the entry of 1 indicates the corresponding element appears in the user's next-period set. We further convert the multi-label classification problem into multiple binary classification problems and optimize the model by minimizing the cross-entropy loss,

$$L = -\sum_{u \in \mathbb{U}} \sum_{v \in \mathbb{V}} Y_{u,v} \log(\hat{Y}_{u,v}) + (1 - Y_{u,v}) \log(1 - \hat{Y}_{u,v}),$$

Model Complexity Analysis

Table 1 shows the space complexity and time complexity of the existing methods and our SFCNTSP in each process.

Space Complexity. The overall space complexity of SFCNTSP is $\mathcal{O}(nc + \bar{t}^2 + c^2)$, which is irrelevant to the number of layers due to the single-layered SFCN backbone. In addition to the embeddings of elements $\boldsymbol{E}$ with $\mathcal{O}(nc)$ space complexity, which are indispensable for all the methods, SFCNTSP just additionally introduces $\boldsymbol{W}_T, \boldsymbol{W}_C$ and $\boldsymbol{W}_P$ with $\mathcal{O}(\bar{t}^2 + c^2)$ space complexity. Therefore, SFCNTSP theoretically has a much lower space complexity than previous methods and can empirically outperform the state-of-the-art with fewer parameters in the experiments.

Time Complexity. The overall time complexity of SFCNTSP is $\mathcal{O}(\bar{t}c(\bar{t} + \bar{n}^2 + \bar{n}c) + nc(\bar{t}\bar{n} + c))$, which is also not affected by the number of layers. The first term denotes the time complexity in learning intra-set temporal dependencies, inter-set element relationships, and intra-embedding channel correlations, which are the main components in our approach based on the efficient SFCNs. The second term is the time complexity in the adaptive fusing of user representations for each element, which effectively enhances the model expressive ability. Overall, our model has lower computation costs than baselines, which will be validated in experiments.

Experiments

In this section, we conduct experiments on four benchmarks to evaluate the effectiveness and efficiency of our approach.

Descriptions of Benchmarks

Following Yu et al. (2022), we use four benchmarks in the experiments, including **JingDong**, **DC**, **TaoBao** and **TMS**.

- **JingDong**[3] records the actions of users about purchasing, browsing, following, commenting, and adding products to shopping carts. The purchasing actions in March 2018

[3] https://jdata.jd.com/html/detail.html?id=8

Datasets	#sets	#users	#elements	#E/S	#S/U
JingDong	15,195	3,063	3,551	1.26	4.96
DC	42,905	9,010	217	1.52	4.76
TaoBao	225,989	49,393	689	1.45	4.58
TMS	243,116	15,726	1,563	2.19	15.46

Table 2: Statistics of the datasets.

Settings	JingDong	DC	TaoBao	TMS
Learning rate	0.001	0.001	0.001	0.001
Dropout rate	0.2	0.1	0.05	0.1
Embedding channels c	64	64	32	64
Hyperparameter α	1.0	1.0	1.0	1.0
Hyperparameter β	0.1	0.1	0.1	0.1

Table 3: Configurations of our approach on all the datasets.

are chosen and products bought on the same day by each user are treated as a set.

- **Dunnhumby-Carbo (DC)**[4] includes the transactions of households at a retailer in two years. Transactions in the first 60 days are selected and products purchased on the same day by each household are treated as a set.
- **TaoBao**[5] contains the online user behaviors about purchasing, clicking, marking products as favors, and adding products to shopping carts. The purchasing behaviors are chosen and the categories of products bought on the same day by each user are treated as a set.
- **Tags-Math-Sx (TMS)**[6] contains the history of users' questions in Mathematics Stack Exchange and we use the preprocessed version in Yu et al. (2022) in experiments.

We strictly follow Yu et al. (2022) to preprocess the datasets. Concretely, we select frequent elements that cover 80% records on JingDong, DC, and TaoBao, and use all the elements on TMS. We drop the sequences with lengths less than 4 and crop the sequences with lengths more than 20. Table 2 shows the statistics of the datasets, where #E/S denotes the average number of elements in each set, and #S/U is the average number of sets of each user.

We evaluate the methods under both transductive and inductive settings. For the transductive setting, we follow Yu et al. (2022) to use the last set, the second last set, and the remaining sets of each user for testing, validation, and training. For the inductive setting, we follow Yu et al. (2020) to randomly split each dataset across users with the ratio of 70%, 10%, and 20% for training, validation, and testing.

Compared Baselines

The following baselines are compared with our approach:

- **TOP** recommends the most frequent elements in the sequences of all the users as the predictions for any user.
- **PerTOP** predicts the most frequent elements in the personalized sequence for each user.
- **FPMC** combines the matrix factorization and Markov chain for next-period basket prediction (Rendle, Freudenthaler, and Schmidt-Thieme 2010).
- **DREAM** first uses pooling operations to represent baskets as vectors, and then feeds the sequence of basket representations into an RNN to predict the next-period basket (Yu et al. 2016).
- **DIN** learns user representations that are specific to each ad by the attention mechanism for click-through rate prediction (Zhou et al. 2018).
- **LightGCN** linearly propagates information on the user-item interaction graph by simplified graph convolutions to learn high-order connectivity (He et al. 2020).
- **MLP4Rec** uses MLP-Mixer (Tolstikhin et al. 2021) as the backbone and proposes a tri-directional fusion scheme for sequential recommendation (Li et al. 2022). We incorporate the proposed permutation-invariant functions into MLP4Rec to adapt it to the temporal sets prediction task. The cross-feature fusion is omitted since there is only one type of feature in our problem.
- **Sets2Sets** first computes the representations of sets by the average pooling and then learns temporal dependencies by the encoder-decoder framework for predicting multi-period sets. The repeated patterns in user behaviors are also considered (Hu and He 2019).
- **DSNTSP** is a dual sequential network, which is built on Transformer and learns both element-level and set-level representations for each user's sequence. A co-transformer module is presented to exploit the temporal dependencies among elements and sets (Sun et al. 2020).
- **DNNTSP** constructs a sequence of element snapshots based on their co-occurrence and then learns on the snapshots with graph convolutions, the attention mechanism, and the gating mechanism. It can also incorporate repeated user behaviors for improving the prediction performance (Yu et al. 2020).
- **ETGNN** first connects the sequences of different users through a temporal graph and then learns on the graph with the element-guided message aggregation mechanism and the temporal information utilization component (Yu et al. 2022).

Experimental Settings

Following Yu et al. (2020, 2022), we rank top-K elements based on the predicted probabilities and set K to 10, 20, 30, and 40 for evaluation. Recall, Normalized Discounted Cumulative Gain (NDCG) and Personal Hit Ratio (PHR) are adopted as the evaluation metrics. We use Adam (Kingma and Ba 2015) as the optimizer with the cosine annealing learning rate scheduler (Loshchilov and Hutter 2017). Dropout (Srivastava et al. 2014) is adopted to prevent models from over-fitting. We train the models for 2000 epochs and use the early stopping strategy with patience of 100. We select the model that achieves the best performance on

[4]https://www.dunnhumby.com/careers/engineering/sourcefiles
[5]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649
[6]https://math.stackexchange.com

Datasets	Methods	K=10			K=20			K=30			K=40		
		Recall	NDCG	PHR	Recall	NDCG	PHR	Recall	NDCG	PHR	Recall	NDCG	PHR
JingDong	TOP	0.1531	0.0988	0.1574	0.1826	0.1076	0.1926	0.2115	0.1143	0.2207	0.2395	0.1198	0.2484
	PerTOP	0.2709	0.2264	0.2905	0.2742	0.2276	0.2935	0.2757	0.2279	0.2954	0.2762	0.2280	0.2964
	FPMC	0.2704	0.2109	0.2880	0.2973	0.2182	0.3134	0.3082	0.2207	0.3245	0.3178	0.2226	0.3346
	DREAM	0.2888	0.2198	0.3033	0.3373	0.2329	0.3513	0.3637	0.2388	0.3787	0.3757	0.2413	0.3918
	DIN	0.3024	0.2503	0.3213	0.3176	0.2545	0.3379	0.3262	0.2565	0.3461	0.3328	0.2580	0.3519
	LightGCN	0.3089	0.2315	0.3265	0.3405	0.2404	0.3595	0.3613	0.2451	0.3810	0.3738	0.2477	0.3934
	MLP4Rec	0.3035	0.2271	0.3173	0.3373	0.2365	0.3523	0.3564	0.2408	0.3715	0.3731	0.2441	0.3875
	Sets2Sets	0.3209	0.2497	0.3418	0.3474	0.2571	0.3696	0.3623	0.2604	0.3843	0.3735	0.2627	0.3960
	DSNTSP	0.3464	0.2734	0.3670	0.3750	0.2820	0.3947	0.3883	0.2852	0.4078	0.3963	0.2869	0.4150
	DNNTSP	0.3224	0.2458	0.3470	0.3568	0.2551	0.3813	0.3747	0.2594	0.3986	0.3843	0.2613	0.4074
	ETGNN	0.3658	0.2724	0.3885	0.4217	0.2878	0.4460	0.4558	0.2956	0.4780	0.4752	0.2997	0.4959
	SFCNTSP	**0.3877**†	**0.2775**†	**0.4070**†	**0.4459**†	**0.2933**†	**0.4652**†	**0.4821**†	**0.3017**†	**0.5003**†	**0.5018**†	**0.3057**†	**0.5191**†
	Improvement	5.99%	1.50%	4.76%	5.74%	1.91%	4.30%	5.77%	2.06%	4.67%	5.60%	2.00%	4.68%
DC	TOP	0.1606	0.0839	0.2326	0.2521	0.1093	0.3430	0.3279	0.1269	0.4251	0.3872	0.1397	0.4906
	PerTOP	0.4080	0.3161	0.5039	0.4383	0.3246	0.5389	0.4636	0.3306	0.5663	0.4982	0.3381	0.5980
	FPMC	0.2462	0.1991	0.3274	0.3175	0.2191	0.4128	0.3771	0.2332	0.4805	0.4323	0.2451	0.5386
	DREAM	0.3159	0.2266	0.4091	0.4102	0.2532	0.5089	0.4813	0.2701	0.5821	0.5427	0.2833	0.6428
	DIN	0.3747	0.2761	0.4752	0.4617	0.3004	0.5673	0.5166	0.3135	0.6222	0.5687	0.3247	0.6717
	LightGCN	0.3412	0.2536	0.4401	0.4358	0.2802	0.5441	0.5066	0.2969	0.6142	0.5590	0.3083	0.6642
	MLP4Rec	0.3730	0.2746	0.4709	0.4769	0.3037	0.5796	0.5479	0.3000	0.6486	0.6061	0.3331	0.7008
	Sets2Sets	0.4417	0.3169	0.5383	0.5031	0.3342	0.6004	0.5533	0.3459	0.6514	0.5936	0.3546	0.6913
	DSNTSP	0.4399	0.3201	0.5386	0.5112	0.3303	0.6105	0.5615	0.3522	0.6608	0.6031	0.3612	0.7004
	DNNTSP	0.4461	0.3176	0.5442	0.5168	0.3374	0.6170	0.5634	0.3483	0.6626	0.6067	0.3575	0.7033
	ETGNN	0.4593	0.3321	0.5582	0.5477	0.3567	0.6454	0.6070	0.3708	0.7009	0.6580	0.3818	0.7468
	SFCNTSP	**0.4734**†	**0.3390**†	**0.5709**†	**0.5653**†	**0.3648**†	**0.6601**†	**0.6264**†	**0.3794**†	**0.7173**†	**0.6755**†	**0.3899**†	**0.7613**†
	Improvement	3.07%	2.08%	2.28%	3.21%	2.27%	2.28%	3.20%	2.32%	2.34%	2.66%	2.12%	1.94%
TaoBao	TOP	0.1572	0.0835	0.1987	0.2457	0.1074	0.2964	0.3091	0.1220	0.3637	0.3609	0.1328	0.4208
	PerTOP	0.1794	0.1240	0.2187	0.1909	0.1272	0.2328	0.1984	0.1289	0.2424	0.2061	0.1305	0.2517
	FPMC	0.1675	0.0959	0.2088	0.2548	0.1196	0.3082	0.3189	0.1343	0.3778	0.3659	0.1440	0.4273
	DREAM	0.1665	0.0932	0.2069	0.2566	0.1177	0.3079	0.3185	0.1319	0.3752	0.3663	0.1419	0.4262
	DIN	0.2188	0.1317	0.2671	0.3056	0.1553	0.3623	0.3646	0.1690	0.4255	0.4088	0.1782	0.4716
	LightGCN	0.1636	0.0951	0.2051	0.2606	0.1214	0.3155	0.3328	0.1380	0.3927	0.3860	0.1491	0.4489
	MLP4Rec	0.2117	0.1243	0.2592	0.3077	0.1504	0.3650	0.3737	0.1656	0.4347	0.4258	0.1764	0.4884
	Sets2Sets	0.2413	0.1488	0.2911	0.3228	0.1710	0.3821	0.3838	0.1850	0.4465	0.4315	0.1950	0.4954
	DSNTSP	0.2363	0.1431	0.2867	0.3296	0.1685	0.3885	0.3932	0.1832	0.4557	0.4414	0.1932	0.5050
	DNNTSP	0.2511	0.1535	0.3028	0.3369	0.1769	0.3972	0.3925	0.1898	0.4535	0.4384	0.1994	0.5024
	ETGNN	**0.2589**	0.1542	0.3103	0.3525	0.1798	0.4134	0.4124	0.1937	0.4760	0.4596	0.2036	0.5239
	SFCNTSP	0.2586	**0.1551**†	**0.3108**	**0.3578**†	**0.1822**†	**0.4188**†	**0.4229**†	**0.1972**†	**0.4872**†	**0.4717**†	**0.2074**†	**0.5367**†
	Improvement	-0.12%	0.58%	0.16%	1.50%	1.33%	1.31%	2.55%	1.81%	2.35%	2.63%	1.87%	2.44%
TMS	TOP	0.2620	0.1658	0.4604	0.3988	0.2076	0.6336	0.4919	0.2319	0.7313	0.5669	0.2502	0.8074
	PerTOP	0.4599	0.3554	0.6535	0.5426	0.3825	0.7315	0.5602	0.3877	0.7458	0.5632	0.3885	0.7480
	FPMC	0.3889	0.3212	0.5817	0.4861	0.3520	0.6910	0.5488	0.3688	0.7548	0.5966	0.3806	0.7984
	DREAM	0.4433	0.3588	0.6391	0.5448	0.3912	0.7393	0.6121	0.4092	0.7991	0.6561	0.4200	0.8338
	DIN	0.4429	0.3339	0.6578	0.5495	0.3675	0.7679	0.6149	0.3849	0.8272	0.6581	0.3954	0.8601
	LightGCN	0.4697	0.3591	0.6972	0.5618	0.3890	0.7767	0.6109	0.4022	0.8157	0.6475	0.4112	0.8442
	MLP4Rec	0.4652	0.3791	0.6841	0.5704	0.4119	0.7790	0.6461	0.4292	0.8321	0.6942	0.4412	0.8710
	Sets2Sets	0.4835	0.3838	0.6924	0.5789	0.4141	0.7834	0.6327	0.4286	0.8284	0.6701	0.4377	0.8586
	DSNTSP	0.4692	0.3609	0.6791	0.5771	0.3951	0.7825	0.6412	0.4122	0.8392	0.6853	0.4230	0.8716
	DNNTSP	0.4861	0.3660	0.6884	0.5905	0.3994	0.7910	0.6504	0.4153	0.8455	0.6930	0.4257	0.8790
	ETGNN	0.5059	0.3914	0.7138	0.6168	0.4267	0.8104	0.6818	0.4442	0.8609	0.7271	0.4553	0.8941
	SFCNTSP	**0.5193**†	**0.4010**†	**0.7185**†	**0.6307**†	**0.4366**†	**0.8180**†	**0.6969**†	**0.4543**†	**0.8701**†	**0.7415**†	**0.4652**†	**0.9009**†
	Improvement	2.65%	2.45%	0.66%	2.25%	2.32%	0.94%	2.21%	2.27%	1.07%	1.98%	2.17%	0.76%

Table 4: Performance of different methods on all the datasets under the transductive setting. The best and second-best performance are boldfaced and underlined. We also show the improvements of our method over the best baseline and use † to denote the improvements are statistically significant via a paired t-test with p <0.05.

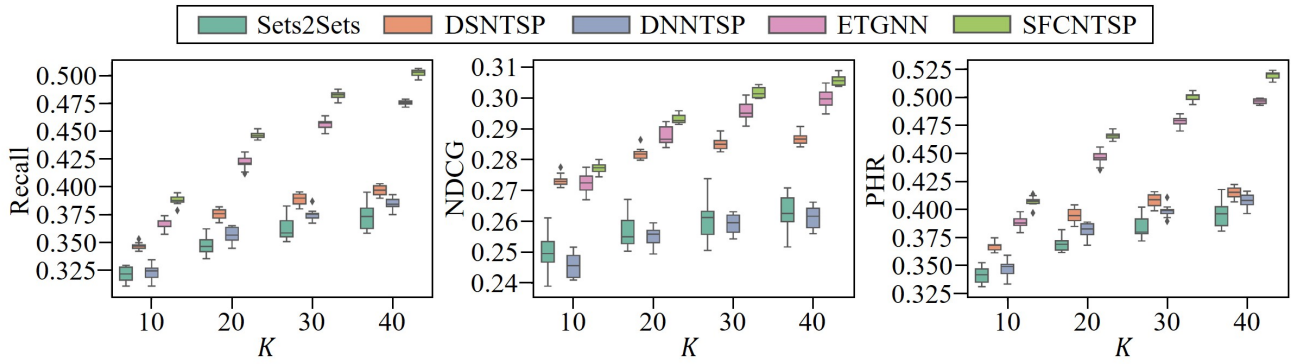

Figure 2: Performance of the baselines and our SFCNTSP over ten runs on JingDong.

the validation set for testing. We set the learning rate and batch size to 0.001 and 64 on all the datasets. We search the dropout rate and the number of embedding channels c in [0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3] and [32, 64, 128]. For hyperparameters α and β, we set α to 1.0 to represent the residual connection and search β in [0.0, 0.01, 0.1, 0.3, 0.5]. The configurations of our model under the transductive setting are shown in Table 3. Under the inductive setting, the configurations are identical except for the dropout rate on JingDong, which is set to 0.25. We run the methods ten times with seeds from 0 to 9 and report the average performance to eliminate deviations. We conduct the experiments on an Ubuntu machine equipped with one Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz with 16 CPU cores. The GPU device is NVIDIA GeForce RTX 3090 with 24 GB memory. Our model is implemented by PyTorch (Paszke et al. 2019).

Performance Under Transductive Setting

The comparisons of all the methods under the transductive setting are shown in Table 4, where the results of baselines are strictly inherited from Yu et al. (2022) with one recent baseline MLP4Rec additionally.

From Table 4, we have the following conclusions: 1) *Learning from user's personalized sequence is essential*. Compared with TOP which indiscriminately predicts the most frequent elements for all the users, PerTOP usually obtains better results as it recommends personalized elements for each user based on the user's own sequence. 2) *It is necessary to leverage the entire historical behaviors*. DREAM often performs better than FPMC because DREAM leverages RNNs to learn temporal dependencies in the whole sequence of each user while FPMC only captures the adjacent behaviors by the Markov chain. 3) *Some insights for recommender systems can also contribute to the prediction of temporal sets*. DIN, LightGCN, and MLP4Rec achieve better performance than the above methods, indicating the advantages of learning element-specific representations (DIN), exploring collaborative signals in high-order connectivity (LightGCN), and mining the sequential and cross-channel relationships (MLP4Rec). However, there is still space for improvement as they fail to capture the properties of temporal sets. 4) *Designing customized methods for temporal sets prediction is important*. Sets2Sets, DSNTSP, DNNTSP, and ETGNN are often superior to other baselines. Sets2Sets additionally models repeated user behaviors. DSNTSP learns both element-level and set-level representations for each user. DNNTSP captures the co-occurrence relationships of elements in each set by GNNs. ETGNN learns element-specific representations which are aware of the temporal information for each user and can explore the collaborative signals in high-order user-element interactions, which is the current state-of-the-art. 5) *SFCNTSP significantly outperforms the existing methods in most cases even if its architecture is quite simple*. The superiority of our approach lies in the learning of inter-set temporal dependencies, intra-set element relationships, and intra-embedding channel correlations. Moreover, the adaptive fusing of user representations further improves the prediction performance.

We further show how we conclude that the improvements of our method are significant over the best baseline. We take the JingDong dataset as an example. Figure 2 shows the performance of the baselines for temporal sets prediction and our approach over ten runs. We observe that SCFNTSP outperforms the baselines with higher median and lower variances. We conduct the paired t-test to compute the statistical significance between the performance of the best baseline and SCFNTSP, where the significance level p is set to 0.05. Specifically, the values of p on Recall/NDCG/PHR of K from 10 to 40 are $4.13e^{-6}$/$5.39e^{-6}$/$2.08e^{-5}$, $1.31e^{-6}$/0.0004/$1.95e^{-5}$, $6.72e^{-7}$/0.0001/$1.76e^{-6}$ and $1.80e^{-9}$/$7.47e^{-5}$/$4.06e^{-9}$. The results show that our method achieves statistically significant improvements over the best baseline with p <0.05.

Performance Under Inductive Setting

We also compare the inductive ability of our method with baselines to evaluate the model performance in predicting new users that are not observed in the training set. We show the performance of different methods on all the datasets under the inductive setting in Figure 3. The results of TOP, FPMC, LightGCN, and ETGNN are not presented because TOP achieves inferior performance, and FPMC, LightGCN, and ETGNN *are inherently not inductive as they are designed with user-specific trainable parameters*.

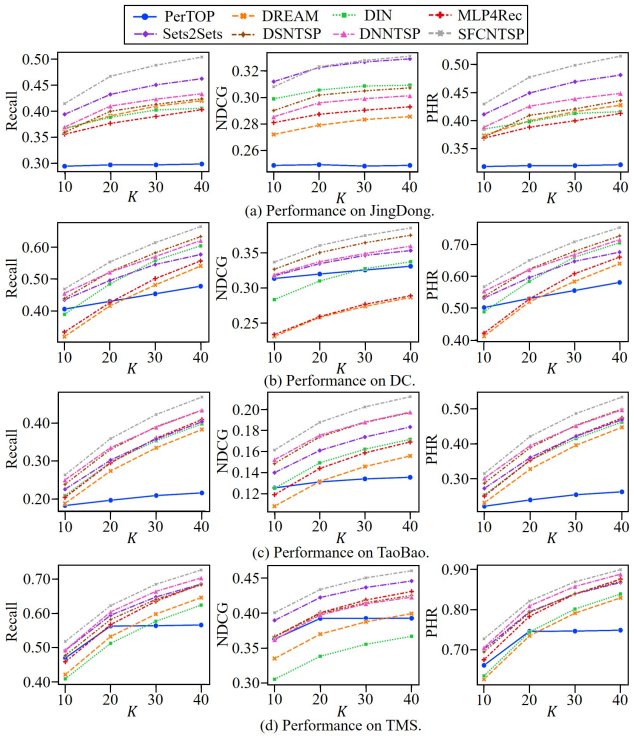

Figure 3: Performance of different methods on all the datasets under the inductive setting.

From Figure 3, we observe that our approach often consistently achieves better performance than baselines under the inductive setting. This may indicate that the lightweight architecture of SFCNTSP can prevent it from overfitting and endow it with a strong generalization ability for new users. So far, we have shown that *temporal sets prediction can be well tackled by a succinct framework without sophisticated components and nonlinear activations*.

Ablation Study

We further validate the effectiveness of the Temporal Dependencies Learning (TDL), Element Relationships Learning (ERL), Channel Correlations Learning (CCL), and User Representations Adaptive Fusing (URAF) components. We respectively remove TDL, ERL, and CCL and we denote the remaining parts as w/o TDL, w/o ERL, and w/o CCL. We replace the adaptive fusing mechanism in URAF by averaging the user representations to obtain a single element-agnostic vectorized representation and denote the variant as w/o URAF. We report the performance of different variants on all the datasets in Figure 4.

From Figure 4, we observe that SFCNTSP achieves the best performance when it uses all the components, and removing any component would lead to worse results. In particular, TDL distinguishes the importance of different sets in the sequence. ERL captures the co-occurrence relationships of elements within the same set. CCL exploits the implicit correlations of multiple embedding channels. URAF enhances the model expressive ability by fusing user representations according to every element. Hence, the contributions of each module are demonstrated.

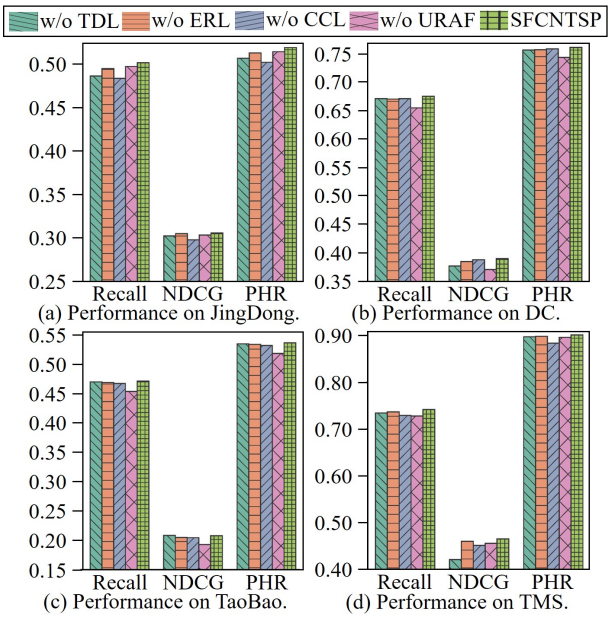

Figure 4: Effects of different components on all the datasets.

Model Efficiency Comparison

We show the efficiency of our approach by comparing the parameter size and running time with baselines for temporal sets prediction. We report the inference time to eliminate the effects of different training strategies of the methods. We show the comparisons of baselines and our SFCNTSP on all the datasets in Figure 5.

Concretely, the parameter size and inference time of SFCNTSP are 0.90MB/0.09MB/0.09MB/0.41MB and 3.45s/8.27s/46.88s/35.54s on JingDong/DC/TaoBao/TMS.

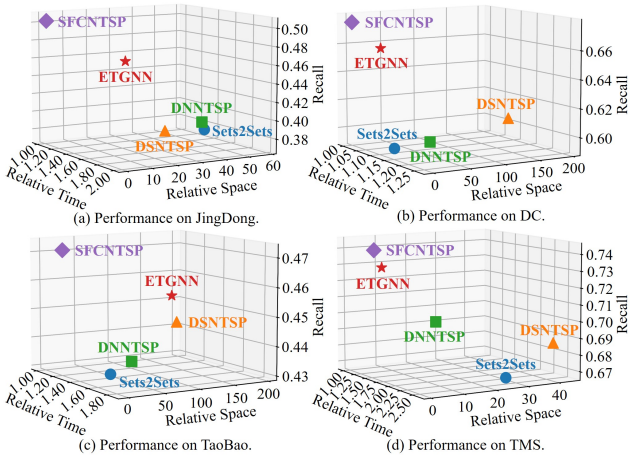

Figure 5: Comparisons of the relative parameter size and inference time of different methods on all the datasets.

Datasets	SFCN	FCN-1	FCN-2	MLP-1	MLP-2
JingDong	**0.5018**	0.4974	0.4941	0.4968	0.4657
	—	-0.88%	-1.53%	-1.00%	-7.19%
DC	0.6755	**0.6760**	0.6744	0.6696	0.6660
	—	0.07%	-0.16%	-0.87%	-1.41%
TaoBao	**0.4717**	0.4715	0.4712	0.4682	0.4654
	—	-0.04%	-0.11%	-0.74%	-1.34%
TMS	0.7415	0.7412	**0.7422**	0.7371	0.7369
	—	-0.04%	0.09%	-0.59%	-0.89%

Table 5: Recall of SFCN, FCN and MLP on all the datasets.

Compared with the baselines, SFCNTSP achieves 33.65×, 58.47×, 79.17×, and 25.77× reductions in space and 1.66×, 1.13×, 1.28×, and 1.22× accelerations in time on average. The significant decreases in space demonstrate the lightweight property of SFCNTSP. For time complexity, though SFCNTSP spends more time on the adaptive fusing of user representations, its overall time complexity is still lower than baselines because of the SCFN backbone.

Comparisons of SFCN with FCN and MLP

We also replace the SFCN with FCN and MLP and compare their performance to validate our assumptions in using SFCN as the backbone. In particular, $ReLU$ is selected as the nonlinear activation function for MLP. We implement the FCN/MLP with one and two hidden layers and denote them as FCN-1/MLP-1 and FCN-2/MLP-2. We report Recall on all the datasets in Table 5, where K is set to 40.

From Table 5, we find that: 1) Compared with FCN, the nonlinear activation function in MLP often leads to worse performance, which validates our first assumption; 2) In FCN, multiple trainable weights are linearly multiplied, which is theoretically identical to multiplying a single trainable weight in SFCN. Therefore, FCN-1 and FCN-2 usually achieve similar performance with SFCN, which is in line with our second assumption; 3) In some cases, stacking more layers may result in overfitting (e.g., FCN-1 and FCN-2 perform worse than SFCN on JingDong, and MLP-2 is inferior to MLP-1 on all the datasets).

Related Work

Temporal Sets Prediction

Temporal sets are defined as a sequence of sets with timestamps, where each set contains an arbitrary number of elements (Benson, Kumar, and Tomkins 2018). The task of temporal sets prediction aims to predict the appearing probabilities of elements in the next-period set. Recently, many methods have be proposed for predicting temporal sets, which can be divided into two types. Methods in the first type followed a two-step strategy with set embedding and sequence learning. For instance, Yu et al. (2016) first obtained the vectorized representation of each basket by pooling operations and then employed RNNs to learn the dynamics in each customer's sequence. Hu and He (2019) and Shi et al. (2021) utilized the average pooling or matrix factorization to compute set representation and captured the temporal dependencies by RNNs and the attention mechanism. Repeated elements in each user's sequence are also modeled. Methods in the second type additionally learned on elements. Sun et al. (2020) jointly derived element-level and set-level representations for each user by a dual sequential network based on Transformer (Vaswani et al. 2017). Yu et al. (2020) combined GNNs, attention and gating mechanisms to learn on the sequence of element snapshots, where each snapshot is constructed by the co-occurrence relationships of elements. Yu et al. (2022) constructed a temporal graph to connect the sequences of different users and then learned element-specific user representations with temporal information for capturing the collaborative signals.

Although insightful, most of the above methods are built on sophisticated components with more trainable parameters and higher computational costs. In this paper, we design a succinct architecture that is solely based on SFCNs for temporal sets prediction.

Simple Architectures in Various Fields

Recently, there is a trend of replacing complicated designs with simple architectures in many fields. For example, in graph learning, Wu et al. (2019) proposed a simplified version of graph convolution networks (Kipf and Welling 2017) by removing the nonlinear activation functions and collapsing the weights between consecutive layers. Rossi et al. (2020) devised a scalable and efficient framework with linear diffusion operators to learn on graphs. In computer vision, Tolstikhin et al. (2021) and Touvron et al. (2021) introduced the MLP-based frameworks for image classification. In recommender systems, He et al. (2020) designed LightGCN to linearly propagate information on the user-item interaction graph by simplified graph convolutions. For sequential recommendations, Zhou et al. (2022) presented FMLP-Rec to replace the multi-head self-attention in Transformer with filter-enhanced MLPs. Inspired by Tolstikhin et al. (2021), Zhou et al. (2022) proposed MLP4Rec with a tri-directional fusion scheme, which is built up by MLPs.

Although the above methods have demonstrated the power of simple architectures in various applications, they are not specialized for temporal sets prediction. In this paper, we aim to specifically design a succinct architecture for predicting temporal sets.

Conclusion

In this paper, we investigated the possibility of designing a succinct architecture for temporal sets prediction. Our approach was solely built on the simplified fully connected networks and could learn inter-set temporal dependencies, intra-set element relationships, and intra-embedding channel correlations with the guarantee of permutation-invariant and permutation-equivariant properties. Experimental results on real-world datasets showed that our approach consistently outperformed the existing baselines under both inductive and transductive settings with fewer trainable parameters and less computational costs. To the best of our knowledge, we are the first to show that a succinct architecture can be competent for temporal sets prediction, which opens up a promising direction for solving this problem.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments on this research. This work was supported in part by the National Key R&D Program of China (2021YFB2104802), and the National Natural Science Foundation of China (62272023 and 51991395).

References

Baytas, I. M.; Xiao, C.; Zhang, X.; Wang, F.; Jain, A. K.; and Zhou, J. 2017. Patient Subtyping via Time-Aware LSTM Networks. In *SIGKDD*, 65–74.

Benson, A. R.; Kumar, R.; and Tomkins, A. 2018. Sequences of Sets. In *SIGKDD*, 1148–1157.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*, 639–648.

Hu, H.; and He, X. 2019. Sets2Sets: Learning from Sequential Sets with Neural Networks. In *SIGKDD*, 1491–1499.

Jin, B.; Yang, H.; Sun, L.; Liu, C.; Qu, Y.; and Tong, J. 2018. A Treatment Engine by Predicting Next-Period Prescriptions. In *SIGKDD*, 1608–1616.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A. R.; Choi, S.; and Teh, Y. W. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML*, volume 97, 3744–3753.

Li, M.; Zhao, X.; Lyu, C.; Zhao, M.; Wu, R.; and Guo, R. 2022. MLP4Rec: A Pure MLP Architecture for Sequential Recommendations. In *IJCAI*, 2138–2144.

Lin, J.; Pu, H.; Li, Y.; and Lian, J. 2017. Intelligent Recommendation System for Course Selection in Smart Education. In *IIKI*, volume 129, 449–453. Elsevier.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 8024–8035.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*, 811–820.

Rossi, E.; Frasca, F.; Chamberlain, B.; Eynard, D.; Bronstein, M. M.; and Monti, F. 2020. SIGN: Scalable Inception Graph Neural Networks. *CoRR*, abs/2004.11198.

Shi, N.; Yu, L.; Sun, L.; Wang, L.; Lin, C.; and Zhang, R. 2021. Deep Heterogeneous Network for Temporal Set Prediction. *Knowl. Based Syst.*, 223: 107039.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958.

Sun, L.; Bai, Y.; Du, B.; Liu, C.; Xiong, H.; and Lv, W. 2020. Dual Sequential Network for Temporal Sets Prediction. In *SIGIR*, 1439–1448.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; Lucic, M.; and Dosovitskiy, A. 2021. MLP-Mixer: An all-MLP Architecture for Vision. In *NeurIPS*, 24261–24272.

Touvron, H.; Bojanowski, P.; Caron, M.; Cord, M.; El-Nouby, A.; Grave, E.; Joulin, A.; Synnaeve, G.; Verbeek, J.; and Jégou, H. 2021. ResMLP: Feedforward networks for image classification with data-efficient training. *CoRR*, abs/2105.03404.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*, 5998–6008.

Wu, F.; Jr., A. H. S.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In *ICML*, volume 97, 6861–6871.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR*, 729–732.

Yu, L.; Sun, L.; Du, B.; Liu, C.; Xiong, H.; and Lv, W. 2020. Predicting Temporal Sets with Deep Neural Networks. In *SIGKDD*, 1083–1091.

Yu, L.; Wu, G.; Sun, L.; Du, B.; and Lv, W. 2022. Element-guided Temporal Graph Representation Learning for Temporal Sets Prediction. In *WWW*, 1902–1913.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Póczos, B.; Salakhutdinov, R.; and Smola, A. J. 2017. Deep Sets. In *NIPS*, 3391–3401.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep Interest Network for Click-Through Rate Prediction. In *SIGKDD*, 1059–1068.

Zhou, K.; Yu, H.; Zhao, W. X.; and Wen, J. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *WWW*, 2388–2399.

Zhu, G.; Wang, Y.; Cao, J.; Bu, Z.; Yang, S.; Liang, W.; and Liu, J. 2021. Neural Attentive Travel package Recommendation via exploiting long-term and short-term behaviors. *Knowl. Based Syst.*, 211: 106511.