Learning Reliable User Representations from Volatile and Sparse Data to Accurately Predict Customer Lifetime Value

Mingzhe Xing[1], Shuqing Bian[2,4], Wayne Xin Zhao[3,4,5*], Zhen Xiao[1*],
Xinji Luo[6], Cunxiang Yin[6], Jing Cai[6], Yancheng He[6]
[1] Department of Computer Science, Peking University, Beijing, China
[2] School of Information, Renmin University of China, Beijing, China
[3] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[4] Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
[5] Beijing Academy of Artificial Intelligence, Beijing, China
[6] Platform and Content Group, Tencent, Shenzhen, China
mzxing@stu.pku.edu.cn,{shuqingbian,batmanfly}@gmail.com,xiaozhen@pku.edu.cn,
{leevenluo,jasonyin,samscai,collinhe}@tencent.com

ABSTRACT

In industry, customer lifetime value (LTV) prediction is a challenging task, since user consumption data is usually volatile, noisy, or sparse. To address these issues, this paper presents a novel Temporal-Structural User Representation (named *TSUR*) network to predict LTV. We utilize historical revenue time series and user attributes to learn both *temporal* and *structural* user representations, respectively. Specifically, the temporal representation is learned with a temporal trend encoder based on a novel multi-channel Discrete Wavelet Transform (DWT) module, while the structural representation is derived with Graph Attention Network (GAT) on an attribute similarity graph. Furthermore, a novel cluster-alignment regularization method is employed to align and enhance these two kinds of representations. In essence, such a fusion way can be considered as the association of temporal and structural representations in the low-pass representation space, which is also useful to prevent the data noise from being transferred across different views. To our knowledge, it is the first time that temporal and structural user representations are jointly learned for LTV prediction. Extensive offline experiments on two large-scale real-world datasets and online A/B tests have shown the superiority of our approach over a number of competitive baselines.

CCS CONCEPTS

• **Human-centered computing → User models**; • **Information systems → Temporal data**.

KEYWORDS

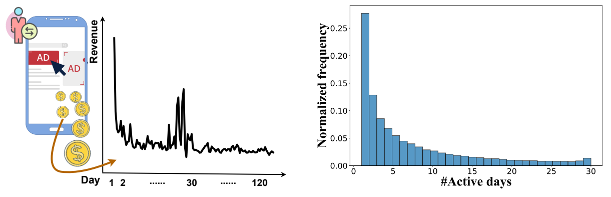
Lifetime Value; Discrete Wavelet Transform; Graph Neural Network

*Corresponding authors.

ACM Reference Format:
Mingzhe Xing[1], Shuqing Bian[2,4], Wayne Xin Zhao[3,4,5*], Zhen Xiao[1*], and Xinji Luo[6], Cunxiang Yin[6], Jing Cai[6], Yancheng He[6]. 2021. Learning Reliable User Representations from Volatile and Sparse Data to Accurately Predict Customer Lifetime Value. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467079

1 INTRODUCTION

As modern economies are becoming predominantly service-based, it is crucial for user-centric companies to build a strong bond between application services and customers. And these companies predict the customer's lifetime value (LTV), a popular metric that measures the value of a user during the lifetime of using an application [11, 34], to reduce user churn and increase retention.



(a) *Volatile* user behavior sequence. (b) *Sparse* user behavior sequence.

Figure 1: Illustration of volatile and sparsity issues in LTV prediction. Here, we present (a) the consumption sequence of a representative user and (b) the distribution of the number of activity days for all the users in our datasets.

In the literature, various LTV prediction methods have been proposed, roughly categorized as probabilistic methods and machine learning based methods [17]. The former category [13, 34] mainly contains the probabilistic generative models for predicting repetitive purchases and customer churn, where the two kinds of behaviors are assumed to follow stochastic process. In the latter category, recent work employs machine learning based methods for LTV prediction [11, 35], which usually rely on hand-crafted features. Recently, deep neural networks such as convolutional neural network [8] are utilized to automatically learn temporal representations for improving LTV prediction.

Although these methods have largely improved the performance of LTV prediction, there are still two major challenges to be solved. Figure 1 illustrates the two cases with real data from our dataset (see Section 5.1.1 for data details). First, as illustrated in Figure 1(a), the consumption history of users is volatile and highly dynamic. Existing methods typically formulate user consumption history as time series, and manually or automatically extract surface features from the sequence data. The extracted features are not able to accurately reflect the stationary characteristics for the underlying trend of user behavior. Second, from Figure 1(b), we can observe that most users are seldom active in most days, which means that the available revenue sequences are often short or sparse. The sparsity problem becomes more severe considering that our goal is to utilize the early stage of customers' lifetime to make prediction for the entire lifetime. Existing LTV prediction studies seldom address this issue, while related studies on user modeling mainly leverage explicit interaction or social graphs to alleviate data sparsity [4, 14], which is usually not available in the setting of LTV prediction.

To address the above issues, the key is to learn reliable user representations for accurate LTV prediction from volatile, sparse behavior sequences. Our solution can be summarized in three major aspects by learning both temporal and structural user presentations. To learn *temporal user representations*, we incorporate the wavelet transform technique [1] to reduce the influence of volatile data by considering low- and high-frequency decomposition. Besides, we utilize auxiliary user attributes to construct an implicit correlation graph, and explicitly learn *structural user representations* to complement and enhance temporal representations. Instead of simply fusing the two kinds of user representations, we borrow the idea in multi-view learning [25, 45] for deriving a more effective representation alignment approach.

To this end, this paper presents a novel Temporal-Structural User Representation (named *TSUR*) model for LTV prediction. We utilize two kinds of data signals (*i.e.,* historical revenue time series and user attributes), to learn user representations in different views, namely temporal and structural user representations. First, we design a novel multi-channel discrete wavelet transform (DWT) [1] module to learn temporal user representations, which is able to capture more reliable temporal features from raw time series. Second, we construct an attribute similarity graph for users and then utilize graph attention network (GAT) [36] to learn structural user representations. Third, we propose a novel cluster-alignment regularization technique to reduce the divergence in the two kinds of user representations. In this way, structural representations can be utilized to refine and enhance temporal representations. In essence, such a fusion way can be considered as the association of temporal and structural representations in the low-pass representation space, which is also useful to prevent the data noise from being transferred across different views. Our final approach fuses the two kinds of user representations for LTV prediction.

To the best of our knowledge, it is the first time that temporal-structural user representations are jointly learned for LTV prediction. Extensive offline experiments on two industry datasets and online A/B tests have shown the effectiveness of our approach for LTV prediction by comparing a number of competitive baselines.

2 RELATED WORK

In this section, we summarize the related works in three aspects.

LTV Prediction. Existing LTV prediction methods can be mainly divided into two categories according to the learning framework. The first category is probability-based methods. The classic BTYD model [34] used customer's transaction history to forecast future LTV. Pareto/NBD [13] combined two different parametric distributions to predict the activity and purchase frequency of users, respectively. However, these probability-based methods heavily rely on prior distribution assumptions. The second category employs machine learning models. Group Random Forest [35] and two-stages XGBoost [11] model were adopted to first identify the premium users, and then to predict their monetary values. Recently, deep neural networks were utilized for LTV prediction. A representative work [8] showed that CNN is superior to MLP in modeling time series data, which is more suitable to LTV prediction.

Time Series Forecast. In another view, LTV prediction can be treated as a special time series forecasting task [8]. Classic approaches include ARIMA [15] and Gaussian process (GP) [32]. These statistical models are simple and interpretable, while they usually make strong assumptions with respect to a stationary process and cannot scale well to multivariate time series data. Deep-learning-based approaches are free from stationary assumptions and they are effective methods to capture non-linearity. For example, MQ-RNN [42] used a local and global RNN module to make stable and robust prediction. To overcome the instability of time series, decomposition-based approaches were proposed to disentangle the original time series into periodical seasonal part and trend in time-based decomposition model [9] and different frequency components in frequency-based decomposition model [47].

Multi-view User Modeling. Multi-view user modeling is a prevalent learning paradigm when user-related data from multiple modalities or domains is available. Elkahky et al. [12] jointly learned features of items from different domains and user features to model users' behavior. Cai et al. [7] designed a joint model to learn the mapping from visual view to text view by simultaneously aligning the two views. Wang et al. [39] considered graphs built from instance-view, category-view, and shop-view, and proposed an inter-view alignment technique to transform information across views.

Our work is built on these studies. While, we are the first to integrate both temporal and structural user representations for LTV prediction. We make important technical contribution on the temporal trend encoder and the multi-view representation alignment.

3 PROBLEM DEFINITION

In this paper, we study the task of predicting customers' future lifetime value (LTV) given their attributes and past consumption behaviors. Formally, for a user u from a user set $\mathcal{U}$, the past consumption behavior is described as a m-length revenue time series, denoted by $\boldsymbol{r}_u = [r_{u,1}, r_{u,2}, r_{u,3}, \dots, r_{u,m}]$, where $r_{u,i}$ is the revenue at the i-th day from user u. Besides, we assume user attributes are also available as input. For user u, let $\boldsymbol{e}_u$ denote the feature vector for u consisting of user attributes, where each entry in $\boldsymbol{e}_u$ corresponds to some specific attributes (either continuous or categorical), such as age and activity degree. Based on the attribute feature vectors,

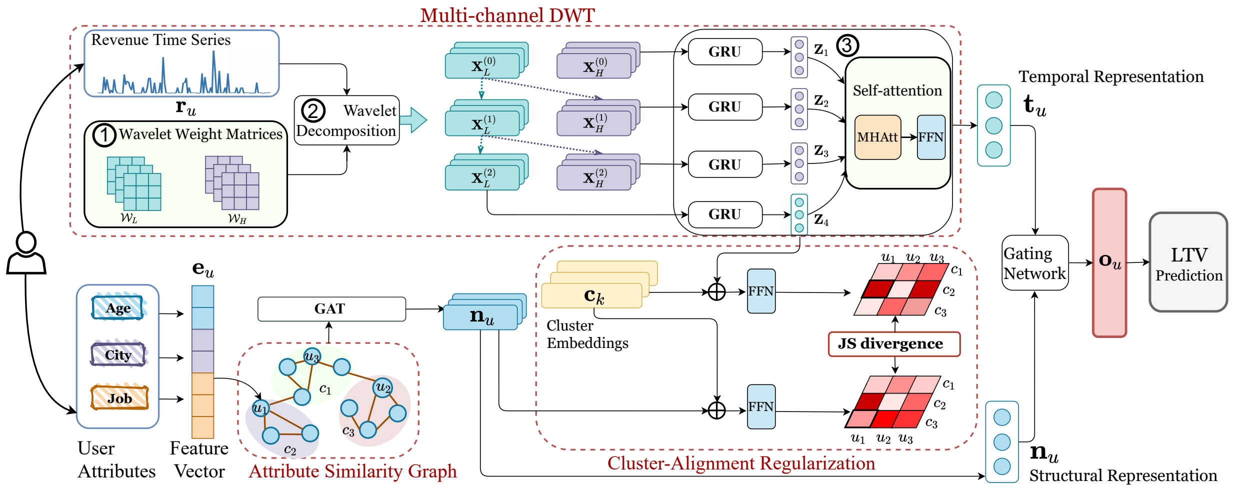


Figure 2: The overall architecture of our proposed TSUR model. "MHAtt" and "FFN" denote multi-head attention and feedforward network, respectively. Modules labeled with ①, ②, ③ correspond to Section 4.1.1, 4.1.2, 4.1.3, respectively.

we can construct an attribute similarity graph, where users are considered as nodes. Given two users u and v, we adopt the Gaussian kernel to measure their similarity: $s_{u,v} = \exp(-\gamma\|\boldsymbol{e}_u - \boldsymbol{e}_v\|^2)$, where γ is a tuning parameter. Following Paredes and Chávez [31], we only keep the top-K most similar neighbors in the graph to reduce the noise of irrelevant links. In this way, we can obtain an adjacency matrix $\mathrm{S} = [s_{u,v}]_{u,v\in\mathcal{U}}$ of size $|\mathcal{U}| \times |\mathcal{U}|$. Revenue time series and attribute similarity graph provide different views to understand the user consumption behavior, either *temporal* or *structural*. We would like to jointly utilize the two views of user data for LTV prediction.

Based on the above notations, given a user u, the corresponding m days historical revenue time series $\boldsymbol{r}_u$ and attribute vector $\boldsymbol{e}_u$, our task is to predict the accumulated LTV for the future Δm days from the $(m+1)$-th day denoted as $y_u = \sum_{i=m+1}^{m+\Delta m+1} r_{u,i}$. In practice, Δm can be tuned according to specific applications.

4 METHOD

In this section, we present a novel Temporal-Structural User Representation (named *TSUR*) model to predict LTV. The core idea is to jointly consider temporal and structural views to derive more reliable user presentations for LTV prediction. In what follows, we first introduce the temporal trend encoder based on a multi-channel discrete wavelet transform (DWT), and then present the graph neural network encoder for attribute similarity graph. Furthermore, we propose a cluster-alignment regularization strategy to effectively align the two views, and combine them for final prediction.

4.1 Temporal Trend Encoder

Due to frequent short-term fluctuations in historical sequences, it is difficult to accurately predict the future consumption propensity based on surface features from raw time series. Considering this difficulty, we design a multi-channel DWT based encoder to learn smooth and reliable underlying temporal consumption trends for LTV prediction. Specifically, inspired by mWDN [38] with a trainable wavelet filter to conduct wavelet decomposition, we first extend the vanilla mWDN by incorporating multi-channel wavelet filters. Then, we perform a novel multi-channel DWT decomposition and obtain multi-channel low- and high-frequency components. Finally, we apply GRU and multi-head attention to model the correlations of channels and frequency components, respectively.

4.1.1 Building Multi-Channel Trainable Wavelet Filters. Wavelet transform is widely used as a multi-resolution denoising approach that produces a smoother series meanwhile maintains its fine structure. It proves to be effective to model and forecast volatile sequence data [1, 33] (more DWT details can be found in Appendix A). Conforming to the homogeneity and orthogonality constraints (Eq. 23, 24, 25 in Appendix A), we initialize the high-pass $\boldsymbol{h} \in \mathbb{R}^a$ and low-pass $\boldsymbol{l} \in \mathbb{R}^a$ wavelet filters. The low- and high-pass weight matrices $\mathbf{W}_L, \mathbf{W}_H \in \mathbb{R}^{m\times m}$ can be defined as follows:

$$\mathbf{W}_L[i, i+j] = \boldsymbol{l}[j], \;\; \mathbf{W}_H[i, i+j] = \boldsymbol{h}[j], \tag{1}$$

where $i, j \in \{1, 2, \cdots, a\}$ and m is the length of time series.

Typically, traditional DWT and mWDN need to pre-set appropriate wavelet filters for a specific time series, which requires prior knowledge or domain experiences. In order to alleviate the bias from pre-set wavelet filters and decompose the original time series in different frequency domains, we design a multi-channel DWT, where each channel corresponds to a low- and high-pass weight matrices constructed by unique low- and high-pass wavelet filters. We can obtain $\mathcal{W}_L = [\mathbf{W}_{L,1}; \mathbf{W}_{L,2}, \cdots, \mathbf{W}_{L,C}]$ and $\mathcal{W}_H = [\mathbf{W}_{H,1}; \mathbf{W}_{H,2}, \cdots, \mathbf{W}_{H,C}] \in \mathbb{R}^{m\times m\times C}$ as the low- and high- weight tensors (see module ① in Figure 2), where C is the number of channels, $\mathbf{W}_{L,c}$ and $\mathbf{W}_{H,c}$ are the low- and high-pass weight matrices for the c-th channel.

4.1.2 Performing Multi-Channel Wavelet Decomposition. Similar to DWT and mWDN, a single-channel DWT decomposes the original m-length time series by multiplying the low- and high-pass weight matrices and then passing the results into average pooling layer to perform the down-sampling:

$$x_L^{(d)} = \text{AvgPool}\big(\sigma(\mathbf{W}_L x_L^{(d-1)} + \boldsymbol{b}_L)\big) \tag{2}$$

$$x_H^{(d)} = \text{AvgPool}\big(\sigma(\mathbf{W}_H x_L^{(d-1)} + \boldsymbol{b}_H)\big), \tag{3}$$

where $x_L^{(d)}, x_H^{(d)} \in \mathbb{R}^{\frac{m}{2^d}}$ are the outputs of low- and high-pass filters at the d-th decomposition level, $x_L^{(0)}$ is set as input time series, and σ is a sigmod activation function, and $\boldsymbol{b}^L$ and $\boldsymbol{b}^H$ are the initial bias.

With $\mathcal{W}_L$ and $\mathcal{W}_H$, we can extend Eq. 2 and 3 in a multi-channel way to produce low- and high-frequency components:

$$\mathbf{X}_L^{(d)} = \text{AvgPool}\big(\sigma(\mathcal{W}_L \mathbf{X}_L^{(d-1)} + \mathbf{B}_L)\big) \tag{4}$$

$$\mathbf{X}_H^{(d)} = \text{AvgPool}\big(\sigma(\mathcal{W}_H \mathbf{X}_L^{(d-1)} + \mathbf{B}_H)\big). \tag{5}$$

where $\mathbf{X}_H^{(d)}=[x_{H,1}^{(d)}; x_{H,2}^{(d)}; \cdots ; x_{H,C}^{(d)}]$, $\mathbf{X}_L^{(d)}=[x_{L,1}^{(d)}; x_{L,2}^{(i)}; \cdots ; x_{L,C}^{(d)}] \in \mathbb{R}^{\frac{m}{2^d} \times C}$ are the high- and low-frequency parts at the d-th decomposition level.

The low-frequency component indicates the underlying long-term consumption trend that dominates the future consumption propensity, while the high-frequency components reflect the unstable short-term consumption tendency (*e.g.,* on Black Friday or shopping festivals). Following Weeks and Bayoumi [40], we keep all the high-frequency components while only reserve the low-frequency component at the last decomposition level to model the future LTV, so we have D+1 frequency components $\{\mathbf{X}_H^{(0)}, \mathbf{X}_H^{(1)}, \cdots, \mathbf{X}_H^{(D)}, \mathbf{X}_L^{(D)}\}$, where D is the number of decomposition levels.

4.1.3 Modeling Correlations for Different Channels and Frequency Components. After obtaining the $D+1$ frequency components, we further utilize a GRU-based layer to characterize the correlations among different wavelet channels and capture the temporal dependencies of original positions. Indeed, a frequency component $\mathbf{X}^{(d)}$ can be regarded as a sub-series that reserves the time order information in the raw time series by a C-dimensional vector. We can feed it into a GRU layer, and derive the hidden state of the last position that it contains: $z = \text{GRU}(\mathbf{X}^{(d)})$. By repeating this process $D+1$ times for all frequency components, we can obtain a matrix consisting of $D+1$ hidden states, denoted by $\mathbf{Z}$:

$$\mathbf{Z} = [z_1; z_2; \cdots ; z_D; z_{D+1}]. \tag{6}$$

To adaptively learn the interaction of different frequency representations, we adopt the multi-head self-attention mechanism on the hidden states $\mathbf{Z}$ (Eq. 6). Specifically, the multi-head self-attention is defined as:

$$\begin{aligned}\mathbf{F}^{(i)} &= \text{MHA}(\mathbf{F}^{(i-1)})\\ &= [head_1, head_2, \cdots, head_h]\mathbf{W}^O \qquad (7)\\ head_j &= \text{Attention}(\mathbf{F}^{(i-1)}\mathbf{W}_j^Q, \mathbf{F}^{(i-1)}\mathbf{W}_j^K, \mathbf{F}^{(i-1)}\mathbf{W}_j^V),\end{aligned}$$

where the $\mathbf{F}^{(i-1)}$ is the input for the i-th layer. When i=1, the input $\mathbf{F}^{(0)}$ is set as $\mathbf{Z}$, and the projection matrix $\mathbf{W}_j^Q$, $\mathbf{W}_j^K$, $\mathbf{W}_j^Q$ and $\mathbf{W}_j^O$ are the corresponding learnable parameters for each attention head. The attention function is implemented by scaled dot-product operation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d/h}})\mathbf{V}. \tag{8}$$

Finally, we take the learned representations $\mathbf{F}$ at the last layer, and apply an average pooling as the *temporal representation* of user u, denoted by $\boldsymbol{t}_u$:

$$\boldsymbol{t}_u = \text{AvgPool}(\mathbf{F}). \tag{9}$$

4.2 Enhancing User Representation by Attribute Similarity Graph Encoder

In our approach, temporal user representations are directly learned from raw time series, which is usually short and sparse. Therefore, we leverage intrinsic similarity of consumption trends between users with similar profiles for enhancing temporal user representations. Specifically, we learn structural user representations on the attribute similarity graph, and associate structural representations with temporal representations based on cluster-alignment regularization.

4.2.1 Attribute Similarity Graph Encoder. Recall that we have constructed the attribute similarity graph with the adjacency matrix S. Based on this similarity graph, we adopt the commonly used graph attention network (GAT) [36] to learn structural user representations by propagating and aggregating node information over graphs. In particular, the node representations can be updated through GAT layer defined as follows:

$$\boldsymbol{p}_i^{(l)} = \mathrm{S}^{(l)}\mathrm{N}_i^{(l)}, \tag{10}$$

$$\boldsymbol{\alpha}_{ij}^{(l)} = \frac{\exp\big(\text{LeakyReLU}(\boldsymbol{v}^T(\boldsymbol{p}_i^{(l)} | \boldsymbol{p}_j^{(l)}))\big)}{\sum_{k\in\mathcal{K}_i}\exp\big(\text{LeakyReLU}(\boldsymbol{v}^T(\boldsymbol{p}_i^{(l)} | \boldsymbol{p}_k^{(l)}))\big)}, \tag{11}$$

$$\mathrm{N}_i^{(l+1)} = \sigma\Bigg(\sum_{j\in\mathcal{K}_i}\boldsymbol{\alpha}_{ij}^{(l)}\boldsymbol{p}_j^{(l)}\Bigg), \tag{12}$$

where S and $\boldsymbol{v}$ are the weight matrix and weight vector, respectively, and $\mathcal{K}_i$ denotes the neighbours set of node i. The representation $\mathrm{N}^{(l-1)}$ will be propagated through the GAT layer to obtain the new representation $\mathrm{N}^{(l)}$, and the input $\mathrm{N}^{(0)}$ of the first GAT layer is the attribute features $\{\boldsymbol{e}_u\}_{u\in\mathcal{U}}$ of users. GAT introduces multi-head attention to enrich the model capacity and to stabilize the learning process. Each attention head is with specific parameters and their outputs can be concatenated as the final representation:

$$\mathrm{N}_i^{(l+1)} = \mathop{\Big|\Big|}_{k=1}^{K} \sigma\Bigg(\sum_{j\in\mathcal{K}_i}\boldsymbol{\alpha}_{ij}^{k}\boldsymbol{p}_j^{(l)}\Bigg), \tag{13}$$

where || represents concatenation, and K is the number of attention heads. Then, we can obtain a latent vector $\boldsymbol{n}_u$ for each user u, called *structural user representations.*

4.2.2 Cluster-Alignment Regularization. The purpose of this step is to effectively align the two kinds of representations for subsequent representation fusion.Since representations in different views encode heterogeneous data characteristics, a simple fusion way might affect the representation capacity in each single view [25, 46]. Following the consensus principle [24, 45], we propose to project both structural representation and temporal representation (low-frequency only) into shared clusters for semantic alignment. Such a cluster-based alignment way can be regarded as a form of low-rank approximation [37], which is able to reduce the noise of each view and derive high-quality representation alignment.

We assume that users are likely to form coherent groups or clusters, where they show similar patterns on consumption behaviors. Assume that there are totally K clusters shared by the two views, and each cluster k is associated with a centroid embedding $\boldsymbol{c}_k$ encoding the main characteristics of this cluster. In each view, we have specific assignments of users to the K cluster centroids according to view-specific user representations. We consider a soft (*i.e.,* probabilistic) assignment of users among the K clusters. Formally, let $\theta_{u,k}$ and $\phi_{u,k}$ denote the probabilities of user u that is assigned to the k-th cluster in the temporal and structural views, respectively. The two kinds of probability assignments are defined as follows:

$$\theta_{u,k} = \frac{(1 + {d^{(1)}_{u,k}}^2/t)^{-\frac{t+1}{2}}}{\sum_{k'}(1 + {d^{(1)}_{u,k}}^2/t)^{-\frac{t+1}{2}}}, \tag{14}$$

$$\phi_{u,k} = \frac{(1 + {d^{(2)}_{u,k}}^2/t)^{-\frac{t+1}{2}}}{\sum_{k'}(1 + {d^{(2)}_{u,k}}^2/t)^{-\frac{t+1}{2}}}, \tag{15}$$

where we compute the assignment probabilities by the Student-t distribution kernel [26] (suitable to situations where the sample size is small and the population standard deviation is unknown), and the distance between user and cluster embeddings $d^{(1)}_{u,k}$ and $d^{(2)}_{u,k}$ in the two views are derived with feedforward layers as:

$$d^{(1)}_{u,k} = \tanh(\mathbf{W}_1[\boldsymbol{z}_{D+1}; \boldsymbol{c}_k] + b_1), \tag{16}$$

$$d^{(2)}_{u,k} = \tanh(\mathbf{W}_2[\boldsymbol{n}_u; \boldsymbol{c}_k] + b_2), \tag{17}$$

where $\boldsymbol{z}_{D+1}$ (Eq. 6) and $\boldsymbol{n}_u$ (Eq. 13) are low-pass temporal representation and structural representation for user u, respectively.

Here we only take the low-frequency temporal representation that present smooth yet fine structure of consumption series. Another note is graph neural networks can also be interpreted as a low-pass filter [6, 29, 43] over graph structure, extracting the global smooth topological features on the whole graph and retaining the commonality of node features [6]. In this way, we indeed associate the two kinds of representations in the low-pass representation space. Formally, we minimize the Jensen-Shannon (JS) divergence [28] between the assignment distributions in two views over all the users as the alignment loss:

$$\begin{aligned}\mathcal{L}_{align} &= \frac{1}{2}KL(\Theta||\frac{\Phi+\Theta}{2}) + \frac{1}{2}KL(\Phi||\frac{\Phi+\Theta}{2}) \\ &= \frac{1}{2}\sum_{u\in\mathcal{U}}\sum_{k=1}^{K}\left(\theta_{u,k}\log\frac{2\theta_{u,k}}{\phi_{u,k}+\theta_{u,k}} + \phi_{u,k}\log\frac{2\phi_{u,k}}{\phi_{u,k}+\theta_{u,k}}\right).\end{aligned} \tag{18}$$

With this loss, the distributions from two views are pulled to close in the cluster-based space. On one hand, it leverages useful information from the structural view for enhancing temporal user representations. On the other hand, it makes the fusion between two views feasible by sharing the same clusters and cluster embeddings. There can be other graph regularization techniques [2, 3]. While, our method seeks to form a looser integration between the two views at cluster level instead of at node level in order to prevent the incorporation of noise across views.

4.3 Optimization and Discussion

Next, we present the overall loss (with two kinds of user representations) to be optimized, and further discuss our model in details.

4.3.1 Temporal-Structural Representation Fusion. For each user u, we can obtain the temporal representation $\boldsymbol{t}_u$ and structural representation $\boldsymbol{n}_u$. After cluster-alignment in Section 4.2.2, we combine the two kinds of representations as the final user representation $\boldsymbol{o}_u$ with a gating mechanism as follows:

$$\begin{aligned}\boldsymbol{o}_u &= g_u \cdot \boldsymbol{q}_u + (1 - g_u) \cdot \boldsymbol{t}_u \\ g_u &= \sigma(\mathbf{W}_2[\boldsymbol{t}_u; \boldsymbol{n}_u] + \boldsymbol{b}_2) \\ \boldsymbol{q}_u &= \tanh(\mathbf{W}_1[\boldsymbol{t}_u; \boldsymbol{n}_u] + \boldsymbol{b}_1).\end{aligned} \tag{19}$$

Then, a linear layer is applied to project $\boldsymbol{o}_u$ for prediction:

$$\hat{y}_u = \mathrm{relu}(\mathbf{W}_3\boldsymbol{o}_u + b_3), \tag{20}$$

where $\hat{y}_u$ is the predicted LTV by our model. We adopt the MSE (Mean Square Error) as the loss function, defined as:

$$\mathcal{L}_{ltv} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}(y_u - \hat{y}_u)^2, \tag{21}$$

where y_u is ground-truth LTV defined in Section 3. And the total loss is the combination of $\mathcal{L}_{ltv}$ and $\mathcal{L}_{align}$:

$$\mathcal{L} = \mathcal{L}_{ltv} + \lambda\mathcal{L}_{align}, \tag{22}$$

where λ is a hyper-parameter to adjust the weight of the alignment loss $\mathcal{L}_{align}$ (Eq. 18). To optimize the total loss, we first separately optimize the temporal trend encoder (Section 4.1) and the structural encoder (Section 4.2.1) by the loss $\mathcal{L}_{ltv}$ in Eq. 21. And, we incorporate the cluster-alignment loss $\mathcal{L}_{align}$ (Eq. 18), and jointly optimize the two encoders via the total loss in Eq. 22. More optimization details can be found in Appendix B.

4.3.2 Discussion. The key of our task is to learn reliable user representations from noisy, sparse data for LTV prediction. Generally, it can be considered as a multi-view learning approach. However, different from existing multi-view user modeling studies [7, 12, 39], we make an important technical contribution. Instead of simply learning feature mapping [7] or sharing [39], we align low-frequency representations that contain smoother and purer information [29, 33] in temporal and structural views (with Eq. 18). In particular, we perform the feature alignment at the cluster level. Such a way can be regarded as a form of low-rank approximation [37], which is helpful to prevent the noise information from being transferred across different views. Since we adopt GAT, a method that aims at learning the local attentions of nodes rather than the global topology [36], we can utilize the model weights learned with existing data to derive representations on new users in an inductive manner [18]. Besides, it is flexible to extend our approach to model the data from more views. For this purpose, the major extensions lie in the incorporation pairwise cluster-alignment loss, where the clusters are shared across all the views.

For service-based applications, we are more concerned about the latency of *online inference*. Given m-day data, the DWT operation from multi-channel can be parallelized with time $O(ma)$, where a is the length of wavelet filter. After that, the GRU and self-attention modules take time $O(m + D^2)$, where D is the DWT decomposition level. For GAT encoder, the structural representation can be obtained with time $O(K)$ from a pre-trained GAT model, where K is the number of neighbors in attribute similarity graph. Indeed, we do not need to compute the regularization part in the inference phrase, so that we can compute the two representations in parallel, which has the time complexity of $\max(O((m+1)a+D^2), O(K))$. As for offline training, the major barrier preventing parallel learning lies in the cluster-alignment regularization, which can be accelerated by *stale synchronous parallel* [19] for parameter update. When building an attribute similarity graph for millions or even more users, we can further adopt *locality sensitive hashing* [10] to accelerate the procedure of finding K nearest neighbors.

5 EXPERIMENTS

In this section, we first setup the experiments, and present the experimental analysis.

5.1 Experimental Setup

5.1.1 Datasets. The two datasets are collected from a real-world application of QQ browser app in Tencent[1] via two major user acquisition channels: (1) pre-installation on new mobile phones and (2) download in app stores, which reflect different app installment purposes, called *PI* and *AS* for short. For each dataset, we randomly sample more than 30,000 anonymous users. Each dataset comprises two kinds of data signals, namely the revenue time series and user attribute information. The monetary value of users is recorded on a daily basis in the form of float number, and the revenue time series spans 120 days. The user attributes contain numerical and categorical features, where the numerical features include age, city level, average session time, maximum session time and the number of user actions, and the categorical features include cooperation agent of user acquisition, gender and mobile phone brand. Table 1 presents the basic statistics of our datasets, including the number of users, average consumption frequency per user and average LTV per user in 120-day data. For evaluation, we split the two datasets into training/validation/test sets with a ratio of $8:1:1$.

Table 1: The statistics of our datasets.

Dataset	#users	average consumption frequency	average LTV
PI	33,505	14.46	2.01
AS	36,264	14.35	2.00

5.1.2 Experiment Settings. To evaluate the performance of our model on different lifetime horizons, based on the first 30-day data, we conduct corresponding experiments on the future 30-day and 90-day LTV, which can be considered as short- and long-term evaluation, respectively. For the parameters in our model, the number of channels and decomposition levels in multi-channel DWT module are set as 50 and 3, respectively. The embedding sizes for GRU and GAT are 100 and 30, respectively, and the number of clusters is 300 and the loss weight λ in Eq. 22 is set as 0.01. We use the Adam optimizer [21] to learn our model, and the learning rate is tuned in $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$. The batch size is empirically set to 256. Early stopping is used with a patience of 5 epochs.

[1]https://www.tencent.com/

5.1.3 Evaluation Metrics and Baselines. Following [11], we adopt Normalized Rooted Mean Square Error (NRMSE) and Normalized Mean Average Error (NMAE) to evaluate the performance of different methods. More metric details can be found in Appendix C.2. Note that the lower these metrics are, the performance is better. And we compare our model with baselines from four categories, namely (1) LTV prediction, (2) time series forecasting, (3) graph neural network and (4) user behavior model, including:

• **Two-stage XGBoost** [11] divides the LTV prediction process into two tasks, churn classification and monetary revenue regression, and utilizes XGBoost to model these two tasks, respectively.

• **Group RandomForest** [35] segments users into groups by consumption frequency, and apply Random Forest to predict LTV for each group separately.

• **WhalesDetector** [8] uses a three-layer CNN with three kernel sizes (7, 3 and 1), respectively, to detect the valuable users.

• **DSANet** [20] utilizes global and local temporal components to capture complex mixtures of global and local features.

• **LSTNet** [22] uses CNN and RNN to extract short-term local dependency and long-term patterns from revenue time series.

• **NBeats** [30] proposes a neural architecture based on backward and forward residual links to model gradually varying trend and recurring seasonality.

• **GraphSAGE** [18] is an inductive method that leverages node attribute information to effectively generate node representations.

• **Graph WaveNet** [44] obtains temporal features with Temporal Convolution Network and then utilizes the temporal representation as node features to analyze the spatial relations with GCN.

• **TiSSA** [23] utilizes the time-interval-based GRU and time-sliced hierarchical self-attention module to exploit both local and global temporal dependency of user behaviors.

For fair comparison, we use both the revenue time series and user attributes as features for all the baselines. For example, we concatenate the original time series with attribute feature vectors to form new features and feed them into WhalesDetector for LTV prediction. For GNN models, we use the same attribute similarity graph as in our approach, and extract the RFM (*recency-frequency-monetary*) [41] features from revenue time series concatenated with user attributes as nodes features. For the baselines, the parameters are set as their default values or tuned on the validation set. To reproduce the results of all the comparison methods, we report their parameter settings in Appendix C. Our code can be accessed via this link: https://github.com/xmzzyo/LTV-prediction. Some modules in baselines, *e.g.,* LSTNet and TiSSA, are implemented by RecBole [48].

5.2 Performance Comparison

We present the results of all the baselines and our model on 30-day and 90-day LTV prediction in Table 2.

First, Group RandomForest learns separate models for users with different consumption frequencies, which predict LTV in an ensemble manner and perform better than Two-stage XGBoost. Second, the WhalesDetector model achieves the best performance among all the baselines. The major reason is that it employs hierarchical convolution operation that can be seen as a smooth process on the revenue time series. Such a technique can eliminate noises to some extent and capture more stationary patterns. Although LSTNet also

Table 2: Performance comparison on next 30-day and 90-day LTV prediction. Best and the second best results are marked in bold and underline. The t-test [5] under NRMSE metric shows that the improvement of our method over the best baseline (WhalesDetector) is significant (p-$value$=0.0022).

Methods	PI				AS			
	30-day		90-day		30-day		90-day	
	NRMSE	NMAE	NRMSE	NMAE	NRMSE	NMAE	NRMSE	NMAE
Two-stage XGBoost	0.8786	0.5709	1.0386	0.6237	0.9012	0.5834	1.0422	0.6275
Group RandomForest	0.6681	0.4625	0.8910	0.5984	0.6853	0.4777	0.8978	0.6107
WhalesDetector	0.5396	0.3167	0.8456	0.4681	0.5467	0.3256	0.8915	0.4935
DSANet	0.7248	0.3619	0.9916	0.5889	0.7273	0.3436	1.0168	0.6302
LSTNet	0.6671	0.3265	0.8860	0.5821	0.7251	0.4075	0.9685	0.6559
NBeats	0.5843	0.3513	0.8834	0.5211	0.5489	0.3392	0.9245	0.5403
GraphSAGE	0.7868	0.5271	0.9886	0.6328	0.7499	0.5101	1.0397	0.6437
Graph WaveNet	0.6266	0.3306	0.9599	0.4482	0.7343	0.4378	0.9582	0.4830
TiSSA	0.7521	0.5478	0.9949	0.7333	0.7756	0.5744	1.0141	0.7311
TSUR (our method)	**0.4274**	**0.2464**	**0.7193**	**0.4220**	**0.4432**	**0.2542**	**0.6863**	**0.3915**

adopts the CNN architecture, it uses a one-layer CNN with a fixed-length convolution kernel, which cannot adaptively discover the temporal patterns at different levels of varying revenue sequences. For time series forecasting models, NBeats performs better than LSTNet and DASNet, since it additionally considers both temporal trend and recurring seasonality. Third, similar to our method, Graph WaveNet models temporal and spatial features simultaneously. It is able to learn effective node representations by considering the relations between the two views and performs better than GraphSAGE, which only learns the structural information. Finally, with time-interval-based GRU and time-sliced hierarchical self-attention to model series segments, TiSSA does not perform well since it is difficult to identify reliable consumption trend for short segments especially when the sequences are sparse.

As a comparison, our model TSUR achieves the best performance on all the metrics across different datasets and tasks. It jointly learns temporal and structural representations from revenue time series and attribute similarity graph. Besides, the two kinds of representations are fused through cluster-alignment regularization, which effectively improves the performance for LTV prediction.

Table 3: Ablation study of our model on LTV prediction.

Future horizon	Variant	PI		AS	
		NRMSE	NMAE	NRMSE	NMAE
30 days	T	0.4660	0.2655	0.4853	0.2792
	S	0.7242	0.4817	0.7304	0.4715
	TS	0.4379	0.2504	0.4594	0.2611
	TSC	**0.4274**	**0.2464**	**0.4432**	**0.2542**
90 days	T	0.7501	0.4434	0.7511	0.4404
	S	0.9847	0.6009	1.0208	0.6957
	TS	0.7448	0.4241	0.7119	0.4026
	TSC	**0.7193**	**0.4220**	**0.6863**	**0.3915**

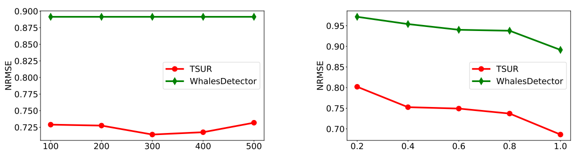
5.3 Ablation Study

We conduct the ablation study to examine effects of different modules in our model. Four variants of our approach are compared, including: **(A)** T denotes using only the temporal representation from multi-channel DWT based trend encoder to predict LTV; **(B)** S denotes using only the structural representation from GAT encoder to predict LTV; **(C)** TS denotes directly fusing the representations from these two encoders to predict LTV; **(D)** TSC denotes our complete model with cluster-alignment regularization.

In Table 3, we can see that the performance order among different variants can be summarized as: S < T < TS < TSC. These results indicate that the temporal trend encoder contributes the most to the final performance, since it can effectively model the reliable trend of revenue time series with a multi-channel DWT based architecture. While, using only structural representations does not perform well. Nevertheless, it can improve the representations of users and make a better prediction by fusing the structural representations. By incorporating the cluster-alignment regularization technique, it further leads to a performance improvement especially for the 90-day prediction, which demonstrates the effectiveness of learning reliable long-term user consumption representation of our model.

5.4 Performance Tuning

In this part, we examine the robustness of our model, and analyze the influence of parameters and training data on model performance. For simplicity, we only incorporate the best baseline WhalesDetector from Table 2 as a comparison.



(a) The number of clusters. (b) The ratio of training data.

Figure 3: Performance tuning for 90-day LTV prediction on AS dataset (results on PI dataset are similar and omitted).

Our model adopts a cluster-alignment mechanism to enhance the temporal representations. The number of clusters K will affect the model performance. We vary K in the set $\{100, 200, 300, 400, 500\}$. It can be observed from Figure 3(a) that $K = 300$ achieves the best performance for our model. While, our model is consistently better than the baseline at the five choices.

Next, we study the performance sensitivity of our model by varying the amount of training data. We take 20%, 40%, 60% and 80% from the complete training data to generate four new training sets, respectively. We fix the test set as original, and then learn the model with new training sets, and report the corresponding results on the test set. As we can see from Figure 3(b), our model consistently outperforms the best baseline from Table 2. Especially, with only 40% training data, our model can achieve a relatively good performance, which improves the baseline by a large margin.

5.5 Case Study

Our model has two major contributions for LTV prediction, namely multi-channel DWT based consumption trend encoder and cluster-alignment regularization. As shown in Table 3, they are useful to improve the model performance. In this part, we show qualitative cases to understand why they work well.

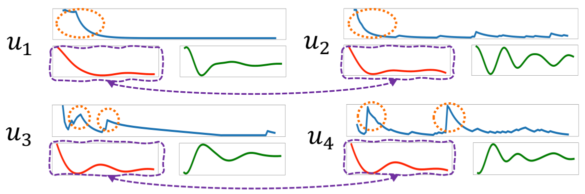


Figure 4: DWT decomposition for four users, where the two pairs correspond to *churn* and *return*, respectively. The raw time series is in blue and the decomposed low- and high-frequency components are in red and green, respectively.

Figure 4 shows the wavelet decomposition results of four sampled users in our datasets, where u_1 and u_2 are churn users and u_3 and u_4 are return users. By reading the original time series, the users in each pair have substantially different shapes, and it is hard to directly identify the difference or connection between two users in a pair. With the wavelet decomposition in our approach, we can see that low-frequency components are indeed very similar for two users in a pair. Recall that low- and high-frequency components correspond to stationary long-term and unstable short-term trends, respectively. This example indicates that our approach is able to reduce the noise or unstable signals from raw time series.

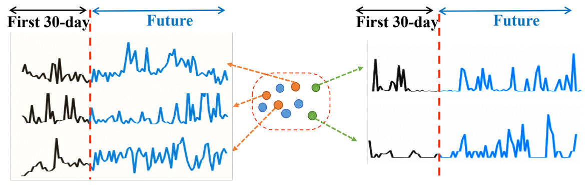


Figure 5: One sampled cluster after cluster-alignment regularization. We only present five users for clarity. The first 30-day and future sequence are in black and blue, respectively.

Furthermore, Figure 5 presents one selected cluster of users after cluster-alignment regularization. With only the first 30-day revenue sequence, the five users tends to be split into two groups (either left or right), since the two users at the right side show a churn tendency of low activity. While, in fact, the two users on the right side are return users with a recurring surge of activity. Based on future data, the five users should be grouped into the same cluster because they share very similar future trends. By inspecting into their profiles, we find this cluster corresponds to users with similar attributes, *e.g.,* young people from second- or third-tier cities. It indicates that structural representations can complement the temporal view and enhance the user representations for LTV prediction.

5.6 Online A/B Test

To further examine the effectiveness of our approach, we conduct online A/B tests in real application scenario of the QQ browser app. Specifically, about 20,000 users are randomly sampled from real traffic, who have a lifetime of at least 30 days (consisting with previous setting) before our experiment. Then, we randomly split them into control group (A) and treatment group (B) with the same size. For comparison, we apply WhalesDetector (the best baseline in Table 2) and our TSUR model to select top n users with the largest predicted LTV from groups A and B, respectively for investment (*i.e.,* profit incentive). Then, we adopt the same investment strategy (omitted to company privacy) to the selected users of the two groups. Finally, we compute the *Return on Investment (ROI)* [16] for both groups, defined as $ROI = \frac{Net\ Return\ on\ Investment}{Cost\ of\ Investment}$. As a commonly used metric to measure the ratio of net profit over a period and cost of investment, a larger ROI value indicates users with potentially higher consumption are found.

Table 4: Comparison of ROI metrics in online A/B test.

Methods	ROI-10	ROI-20
WhalesDetector	0.1420	0.3571
TSUR	**0.1636**	**0.3699**

Here, we perform the comparison with 10-day prediction and 20-day prediction. Table 4 presents the ROI comparison between WhalesDetector and our method TSUR. As we can see, our model TSUR is consistently better than the compared baseline, which further demonstrates the effectiveness of the proposed model.

6 CONCLUSION

In this paper, we proposed a Temporal-Structural user representation model for LTV prediction. We considered two kinds of data inputs. For temporal trend encoder, we developed an improved multi-channel DWT to learn more reliable temporal user representations. For structural encoder, we leveraged GAT to learn structural user representations over attribute similarity graph. In particular, a novel cluster-alignment regularization technique was proposed to align the two kinds of user representations. Extensive offline experiments on two real-world datasets and online A/B tests have shown the effectiveness of our approach.

Currently, we only decompose the revenue time series with respect to frequency domain. We will incorporate other influencing factors such as bursty social events in future work. Besides, we will consider leveraging other kinds of user correlation data such as social graphs to learn better structural user representations.

ACKNOWLEDGEMENT

This work was supported by the National Key R&D Program of China under Grant No. 2020AAA0105200, National Natural Science

Foundation of China under Grant No. 61872397 and 61872369, and Beijing Academy of Artificial Intelligence (BAAI) under Grant No. BAAI2020ZJ0301. Xin Zhao and Zhen Xiao are the corresponding authors.

REFERENCES

[1] Rumaih M Alrumaih and Mohammad A Al-Fawzan. 2002. Time series forecasting using wavelet denoising an application to Saudi Stock Index. *Journal of King Saud University-Engineering Sciences* 14, 2 (2002), 221–233.

[2] Rie K Ando and Tong Zhang. 2007. Learning on graph with Laplacian regularization. In *Advances in neural information processing systems*. 25–32.

[3] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. 2004. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*. Springer, 624–638.

[4] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[5] Bhaskar Bhattacharya and Desale Habtzghi. 2002. Median of the p value under the alternative hypothesis. *The American Statistician* 56, 3 (2002), 202–206.

[6] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. *arXiv preprint arXiv:2101.00797* (2021).

[7] Jia-Jia Cai, Jun Tang, Qing-Guo Chen, Yao Hu, Xiaobo Wang, and Sheng-Jun Huang. 2019. Multi-View Active Learning for Video Recommendation.. In *IJCAI*. 2053–2059.

[8] Pei Pei Chen, Anna Guitart, Ana Fernández del Río, and Africa Periánez. 2018. Customer lifetime value in video games using deep learning and parametric models. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2134–2140.

[9] Robert B Cleveland, William S Cleveland, Jean E Mcrae, and Irma Terpenning. 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Stats* 6, 1 (1990).

[10] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. 2011. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1073–1081.

[11] Anders Drachen, Mari Pastor, Aron Liu, Dylan Jack Fontaine, Yuan Chang, Julian Runge, Rafet Sifa, and Diego Klabjan. 2018. To be or not to be... social: Incorporating simple social features in mobile game customer lifetime value predictions. In *Proceedings of the Australasian Computer Science Week Multiconference*. 1–10.

[12] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*. 278–288.

[13] Peter S Fader, Bruce GS Hardie, and Ka Lok Lee. 2005. RFM and CLV: Using iso-value curves for customer base analysis. *Journal of marketing research* 42, 4 (2005), 415–430.

[14] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[15] Z Asha Farhath, B Arputhamary, and L Arockiam. 2016. A Survey on ARIMA Forecasting Using Time Series Model. *Int. J. Comput. Sci. Mobile Comput* 5 (2016), 104–109.

[16] George T Friedlob and Franklin J Plewa Jr. 1996. *Understanding return on investment*. John Wiley & Sons.

[17] S Gupta, D Hanssens, B Hardie, W Kahn, V Kumar, N Lin, N Ravishanker, and S Sriram. 2016. Modeling Customer Lifetime Value. *Journal of Service Research* 9, 2 (2016), 139–155.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[19] Qirong Ho, James Cipar, Henggang Cui, Jin Kyu Kim, Seunghak Lee, Phillip B Gibbons, Garth A Gibson, Gregory R Ganger, and Eric P Xing. 2013. More effective distributed ml via a stale synchronous parallel parameter server. *Advances in neural information processing systems* 2013 (2013), 1223.

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

[21] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

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

[23] Chenyi Lei, Shouling Ji, and Zhao Li. 2019. Tissa: A time slice self-attention approach for modeling sequential user behaviors. In *The World Wide Web Conference*. 2964–2970.

[24] Hongfu Liu and Yun Fu. 2018. Consensus guided multi-view clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 4 (2018), 1–21.

[25] Khanh Luong and Richi Nayak. 2020. A Novel Approach to Learning Consensus and Complementary Information for Multi-View Data Clustering. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 865–876.

[26] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[27] Stephane G Mallat. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence* 11, 7 (1989), 674–693.

[28] ML Menéndez, JA Pardo, L Pardo, and MC Pardo. 1997. The jensen-shannon divergence. *Journal of the Franklin Institute* 334, 2 (1997), 307–318.

[29] Hoang NT and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[30] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.

[31] Rodrigo Paredes and Edgar Chávez. 2005. Using the k-Nearest Neighbor Graph for Proximity Searching in Metric Spaces. (2005).

[32] Carl Edward Rasmussen. 2003. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 63–71.

[33] Manel Rhif, Ali Ben Abbes, Imed Riadh Farah, Beatriz Martínez, and Yanfang Sang. 2019. Wavelet transform application for/in non-stationary time-series analysis: a review. *Applied Sciences* 9, 7 (2019), 1345.

[34] David C Schmittlein, Donald G Morrison, and Richard Colombo. 1987. Counting your customers: Who-are they and what will they do next? *Management science* 33, 1 (1987), 1–24.

[35] Ali Vanderveld, Addhyan Pandey, Angela Han, and Rajesh Parekh. 2016. An engagement-based customer lifetime value system for e-commerce. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 293–302.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[37] Dong Wang, Qiyue Yin, Ran He, Liang Wang, and Tieniu Tan. 2015. Multi-view clustering via structured low-rank representation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1911–1914.

[38] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2437–2446.

[39] Menghan Wang, Yujie Lin, Guli Lin, Keping Yang, and Xiao-ming Wu. 2020. M2GRL: A Multi-task Multi-view Graph Representation Learning Framework for Web-scale Recommender Systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2349–2358.

[40] Michael Weeks and Magdy Bayoumi. 2003. Discrete wavelet transform: architectures, design and performance issues. *Journal of VLSI signal processing systems for signal, image and video technology* 35, 2 (2003), 155–178.

[41] Jo-Ting Wei, Shih-Yen Lin, and Hsin-Hung Wu. 2010. A review of the application of RFM model. *African Journal of Business Management* 4, 19 (2010), 4199–4206.

[42] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053* (2017).

[43] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[44] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 1907–1913.

[45] Xijiong Xie and Shiliang Sun. 2019. Multi-view support vector machines with the consensus and complementarity information. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[46] Yan Yang and Hao Wang. 2018. Multi-view clustering: A survey. *Big Data Mining and Analytics* 1, 2 (2018), 83–107.

[47] Shuochao Yao, Ailing Piao, Wenjun Jiang, Yiran Zhao, Huajie Shao, Shengzhong Liu, Dongxin Liu, Jinyang Li, Tianshi Wang, Shaohan Hu, et al. 2019. Stfnets: Learning sensing signals from the time-frequency perspective with short-time fourier neural networks. In *The World Wide Web Conference*. 2192–2202.

[48] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *arXiv preprint arXiv:2011.01731* (2020).

APPENDIX

We offer some technical details and reproducibility-related information as supplementary materials to help readers understand and reproduce our model.

A DISCRETE WAVELET TRANSFORM

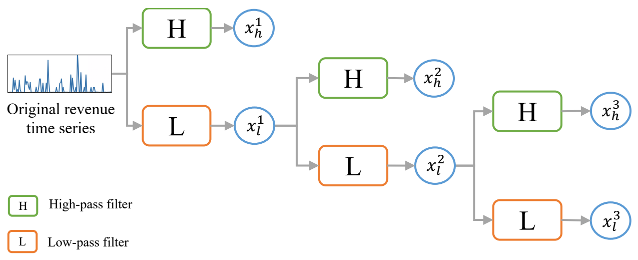


Figure 6: Three levels Mallat DWT process. After input the original revenue time series, the DWT produces different frequency sub-series (in blue circles) $\mathcal{X} = \{x_h^0, x_h^1, \cdots, x_h^n, x_l^n\}$ with low-pass and high-pass wavelet filters respectively.

Discrete Wavelet Transform (DWT) [40] is any wavelet transform for which the wavelets are discretely sampled, which has good temporal resolution by capturing both frequency and location information from time series. The DWT has been successfully applied over a wide range of fields in order to decompose the non-stationary time series into time-frequency domain. Multiresolution denoising using wavelet transform produces a smoother series yet maintains the fine structure of that series. It proves to be very beneficial to series modeling and forecasting [1, 33].

To perform multi-resolution DWT, an efficient way is through the Mallat algorithm [27], which passes the origin time series though a low-pass filter $\boldsymbol{l} = [l_1, l_2, \cdots, l_a]$ and a high-pass filter $\boldsymbol{h} = [h_1, h_2, \cdots, h_a]$, where the low- and high-pass filters conform the following constrains:

Constrain 1: Homogeneity:

$$\sum_a l_i = \sqrt{2} \tag{23}$$

Constrain 2: Orthogonality

$$\sum_a l_i l_{i+2m} = \sigma_{m,0} \tag{24}$$

$$\sum_a h_i h_{i+2m} = \sigma_{m,0}. \tag{25}$$

The low-frequency and high-frequency components can be obtained as follow:

$$\boldsymbol{x}_l^n = \sum_{t=0}^{T-1} \boldsymbol{l}[2T-t]\boldsymbol{x}_l^{n-1}[t] \tag{26}$$

$$\boldsymbol{x}_h^n = \sum_{t=0}^{T-1} \boldsymbol{h}[2T-t]\boldsymbol{x}_h^{n-1}[t], \tag{27}$$

where $\boldsymbol{x}_l^n$ and $\boldsymbol{x}_h^n$ are the outputs of low-pass and high-pass filters at the n-th decomposition level, named approximation and detail coefficients, or low-frequency and high-frequency component respectively, and $\boldsymbol{x}_l^0$ is the original time series when n=1. As illustrated in Figure 6, the decomposition process is iteratively applied on the low-frequency part at different levels to get $\mathcal{X}(n) = \{\boldsymbol{x}_h^0, \boldsymbol{x}_h^1, \cdots, \boldsymbol{x}_h^n, \boldsymbol{x}_l^n\}$, called as the n-th level decomposed results.

B LEARNING ALGORITHM

Algorithm 1 presents the training algorithm for our TSUR model. The entire procedure of our approach consists of three important modules, namely temporal trend encoder, attribute similarity graph encoder and cluster-alignment regularization module.

We first initialize trainable wavelet filters for multi-channel DWT conforming to the homogeneity and orthogonality constraints (Eq. 23, 24, 25) and build attribute similarity graph based on attribute vectors (line 2-3). By conducting multi-channel DWT and modeling the correlations for different channels and frequency components (line 5-6), we can obtain the temporal user representation (line 7). For line 8, the structural user representation is derived with GAT encoder. After the temporal and structural user representations are learned, we align the two representations by cluster-alignment regularization (line 10). Finally, we make LTV prediction (line 12) with the fused user representation obtained in line 11.

Algorithm 1 The training algorithm for the TSUR model.

```
Input: Revenue time series $\{\boldsymbol{r}_u\}$ and user attribute vector $\{\boldsymbol{e}_u\}$ for the users in $\mathcal{U}$.
Output: The predicted LTV $\{\hat{y}_u\}_{u\in\mathcal{U}}$.
1: Randomly initialize the parameters in the TSUR model;
2: Initialize multi-channel wavelet filters conforming to constraints (Eq. 23, 24, 25);
3: Build attribute similarity graph with attribute vectors;
4: for $u = 1 \rightarrow |\mathcal{U}|$ do
5:     Perform multi-channel wavelet decomposition by Eq. 4, 5.
6:     Model correlations for channels and frequency components with GRU and self-attention as introduced in Section 4.1.3.
7:     Obtain temporal user representation $\boldsymbol{t}_u$ by Eq. 9.
8:     Learn structural user representations $\boldsymbol{n}_u$ with the GAT encoder by Eq. 13.
9:     Compute cluster assignment probabilities for $\boldsymbol{t}_u$ and $\boldsymbol{n}_u$ by Eq. 14, 15.
10:    Calculate cluster-alignment regularization loss by Eq. 18.
11:    Obtain fused representation for user $u$ by Eq. 19.
12:    Predict LTV based on fused user representation by Eq. 20.
13:    Calculate total loss by Eq. 22 and optimize model parameters by Adam optimizer.
14: end for
```

C ADDITIONAL EXPERIMENTS DETAILS

C.1 Parameter settings

In this part, we will introduce the parameter settings of baselines. For the baselines, all the models have some parameters to be tuned. We either follow the reported optimal parameter settings or optimize each model separately using the validation set. We report the parameter settings used throughout the experiments in Table 5.

Table 5: Parameter settings of baselines.

Models	Settings

Group RandomForest	max_depth=50
	n_estimators=50
	min_samples_split=0.5

Two-stage XGBoost	learning_rate=0.005,
	max_depth=100,
	n_estimators=100

WhalesDetector	kernels_size=[7, 3, 1],
	channel_num=[50, 100, 50],
	learning_rate=0.00001
	batch_size=256,
	Adam optimizer

LSTNet	RNN_hidden_dim=100,
	CNN_hidden_dim=100,
	kernel_size=3,
	learning_rate=0.00001,
	batch_size=256,
	Adam optimizer

DSANet	attention_head_num=4,
	kernel_size=10,
	learning_rate=0.0001,
	batch_size=256,
	Adam optimizer

NBeats	hidden_dim=100,
	blocks_num=5,
	learning_rate=0.00001,
	batch_size=256,
	Adam optimizer

Graph WaveNet	kernel_size=2,
	blocks_num=4,
	channel_num=512,
	learning_rate=0.00001,
	batch_size=256,
	Adam optimizer

GraphSAGE	hidden_dim=100,
	n_layers=3,
	learning_rate=0.0001,
	batch_size=256,
	Adam optimizer

TiSSA	RNN_hidden_dim=100,
	attention_head=4,
	learning_rate=0.00001,
	batch_size=256,
	Adam optimizer

C.2 Evaluation Metrics

In this part, we introduce the details about the Normalized Rooted Mean Square Error (NRMSE) and Normalized Mean Average Error (NMAE) evaluation metrics.

NRMSE is the fraction of rooted mean square error and mean of ground truth. NRMSE can be computed as:

$$\mathrm{NRMSE} = \frac{\sqrt{\sum_{i=1}^{|\mathcal{U}|} ||\hat{y}_i - y_i||^2} \times \sqrt{|\mathcal{U}|}}{\sum_{i=1}^{|\mathcal{U}|} y_i} \tag{28}$$

NMAE is the fraction of mean absolute error and mean of ground truth. NMAE can be computed as:

$$\mathrm{NMAE} = \frac{\sum_{i=1}^{|\mathcal{U}|} |\hat{y}_i - y_i|}{\sum_{i=1}^{|\mathcal{U}|} y_i} \tag{29}$$

,