Representation Learning on Variable Length and Incomplete Wearable-Sensory Time Series

XIAN WU, CHAO HUANG, PABLO ROBLES-GRANDA, and NITESH V. CHAWLA, University of Notre Dame

The prevalence of wearable sensors (e.g., smart wristband) is creating unprecedented opportunities to not only inform health and wellness states of individuals, but also assess and infer personal attributes, including demographic and personality attributes. However, the data captured from wearables, such as heart rate or number of steps, present two key challenges: (1) the time series is often of variable length and incomplete due to different data collection periods (e.g., wearing behavior varies by person); and (2) there is inter-individual variability to external factors like stress and environment. This article addresses these challenges and brings us closer to the potential of personalized insights about an individual, taking the leap from quantified self to qualified self. Specifically, HeartSpace proposed in this article learns embedding of the time-series data with variable length and missing values via the integration of a time-series encoding module and a pattern aggregation network. Additionally, HeartSpace implements a Siamese-triplet network to optimize representations by jointly capturing intra- and inter-series correlations during the embedding learning process. The empirical evaluation over two different real-world data presents significant performance gains over state-of-the-art baselines in a variety of applications, including user identification, personality prediction, demographics inference, job performance prediction, and sleep duration estimation.

CCS Concepts: • Information systems → Information systems applications;

Additional Key Words and Phrases: Representation learning, wearable-sensory time series

1 INTRODUCTION

The wide proliferation of wearables and mobile devices is revolutionizing health and wellness with the potential of data and personalized insights at one’s fingertips [28]. These wearables generate chronologically ordered streams (e.g., the series of heart rate measurements) or point-in-time activity (e.g., the number of steps) or general summarization of the day (e.g., move goals). Collectively, these data provide an unprecedented opportunity to learn about health and wellness states, as well
as how those interact with the social network, opinions, beliefs, personality, and job performance. In this article, we specifically focus on the chronologically ordered data (such as the heart rate data), which we call wearable-sensory time series. It is important to be able to effectively model these wearable-sensory time-series data for their perceived benefit for a wide spectrum of applications, such as personality detection [6], job performance prediction, health and wellness state assessment and prediction, user identification [40], and demographics inference [41]. However, wearable-sensory time series comes with a number of challenges, including temporal dependencies, incompleteness, intra-sensor/individual variability, and inter-individual variability. To solve these challenges, the fundamental question is how do we effectively featureize the wearable-sensory time-series data, while achieving the generalization purpose for a multitude of applications?

The contemporary body of research in time series has focused on wavelet-based frequency analysis [35] and motif discovery methods [45]. But, these methods carry their own limitations. First, while we can extract discriminating and independent features using wavelet decomposition approaches, it still involves manual efforts and domain-specific expert knowledge (e.g., medical knowledge) [2]. Second, discovering motifs are computationally expensive and require the repeated process of searching for optimal motifs from candidates [18].

Motivated by these limitations and the success of representation learning for automating feature discovery, recent research has led to representation learning on time-series data [3, 26]. However, these works have focused on representation learning for fixed-length sequential data and face their own set of challenges. To learn effective representations for wearable-sensory time-series data, simply applying these methods faces the following key challenges:

- **(C1) Handling variable-length time series and data incompleteness.** A straightforward way to address variable-length input is to resize each input time series into a fixed-length vector. However, the resulting vector, derived by interpolation when input length differs from pre-defined length, has less control of variable time-series resolutions and fails to capture the consistent time granularity for different individuals, leading to inferior representation. An alternative approach is sequence padding with specific symbols so that all time series are as long as the longest series in the dataset [33]. However, an associated challenge with sequence padding is that all time-series data have been artificially created with fixed-length data inefficiently [10]. Consequently, manually created sequences with padding operations may not properly reflect their time-evolving distributions. This is especially critical when trying to draw insights from physiological response data such as heart rate. Thus, a representation learning method should be able to deal with variable length and incomplete data.

- **(C2) Differentiating unique patterns from time-series data with similar trends.** The wearable-sensory data collected from different people may have very similar distributions [42]. For example, for adults 18 and older, a normal resting heart rate (i.e., number of heart beats per minute) is between 60 and 80 beats per minute, and for children ages 6 to 15, the normal resting heart rate is between 70 and 100 [3]. However, there are localized patterns among individuals (intra-sensor/individual variability) that offer important nuances on similarities and differences. A promising representation learning method should be capable of capturing the global and local characteristics of such time-series data to effectively compute similarities among individuals.

In light of these challenges and limitations, we developed a time-series representation learning model, HeartSpace, that addresses the weaknesses of the aforementioned methods and learns generalized and effective embeddings. HeartSpace can benefit a variety of downstream applications, including inferring and predicting user demographics, personality, work performance attributes, and health states. Specifically, in HeartSpace, we first segment the heart rate data collected from
individuals into day-long time series because human behavior presents day-long regularities. Then, a deep autoencoder architecture is developed to map high-dimensional time-specific (i.e., daily) sensor data into the same latent space with a dual-stage gating mechanism. These learned embedding vectors are capable of not only preserving temporal variation patterns, but also handling the data missing issue of wearable-sensory time series well. After that, we leverage a temporal pattern aggregation network to capture inter-dependencies across time-specific representations based on the developed position-aware multi-head attention mechanism. During the training process, a Siamese-triplet network optimization strategy is designed to retain the relations of implicit intra-series temporal consistency and inter-series discrepancy. To summarize, our major contributions are as follows:

- To the best of our knowledge, we are the first to perform a comprehensive study on addressing challenges of learning representations on variable length and incomplete wearable time-series data.
- We develop an unsupervised learning framework, HeartSpace, to learn a generalizable low-dimensional and latent embedding by aggregating variable-length time series and capturing their intra-series and inter-series relations. HeartSpace is also flexible to be generalized into semi-supervised learning for various heart rate analysis application.
- We evaluate HeartSpace on two real-world wearable-sensory time-series data representing a diverse group of human subjects and two different sensing devices. Our experiments demonstrate that HeartSpace outperforms the state-of-the-art methods in various applications, including user identification, personality prediction, demographics inference, job performance prediction, and sleep duration estimation, thereby demonstrating the effectiveness of the learned embeddings.

2 PROBLEM FORMULATION

We first introduce preliminary definitions and formalize the problem of wearable-sensory time-series representation learning. We use bold capital and bold lowercase letters to denote matrices and vectors, respectively.

Definition 1 (Wearable-Sensory Time Series). Suppose there are $I$ users ($U = \{u_1, \ldots, u_i, \ldots, u_I\}$), and $\mathbb{R}^{J_i} X_i = (x_{1i}, \ldots, x_{ji}, \ldots, x_{Jki}) (X_i \in \mathbb{R}^{J_i})$ to denote the series of temporally ordered wearable-sensory data (e.g., heart rate measurement—the number of times a person’s heart beats per minute) of length $J_i$ collected from user $u_i$. In particular, each element $x_{ji}$ represents the $j$-th measured quantitative value from user $u_i (1 \leq j \leq J_i)$ and $J_i$ may vary from user to user. Each measurement $x_{ji}$ is associated with a timestamp information $t_{ji}$ and thus we define a time vector $T_i = (t_{1i}, \ldots, t_{ji}, \ldots, t_{Jki})$ to record the timestamp information of sequential wearable sensor data $X_i$.

In reality, since the wearable-sensory data is collected over different time periods (e.g., with different start and end dates), the sequence lengths usually vary among different time series [19]. Additionally, readings of wearable sensors are usually lost at various unexpected moments because of sensor or communication errors (e.g., power outages) [22]. Therefore, the wearable-sensory data often exhibits variable-length and missing values.

Problem Statement (Wearable-Sensory Time-Series Representation Learning). Given a wearable-sensory time series $X_i$ with variable length and missing values, the objective is to learn the $d$-dimensional latent vector $Y_i \in \mathbb{R}^d$ that is able to capture the unique temporal patterns of time series $X_i$.

The output of the model is a low-dimensional vector $Y_i$ corresponding to the latent representation of each wearable-sensory time series $X_i$ from user $u_i$. Notice that, although different time
series $X_i$ ($i \in [1, \ldots, I]$) can be of any length, their representations are mapped into the same latent space. This learned time-series representation, generated by HeartSpace, can benefit various health and wellness tasks without the requirement of substantial training instance pairs.

3 METHODOLOGY: HEARTSPACE

Time-Series Segmentation. Considering that periodicity has been demonstrated as an important factor that governs human sensory data (e.g., heart rate) with time-dependent transition regularities [25], and the sensed time-series data are often variable length, we first partition the wearable-sensory time series (i.e., $x_i$) of each user $u_i$ into $T$ (indexed by $t$) separated day-long time series ($T$ may vary among users).

Definition 2 (Day-long Time Series $x_{i,t}$). Each $t$-th divided day-long time series of $x_i$ is denoted as $x_{i,t} \in \mathbb{R}^K$, where $K = 1,440$ is the number of minutes included in 1 day. In $x_{i,t}$, each element $x_{i,t}^k$ is the measurement from user $u_i$ at the $k$-th timestep in $x_{i,t}$. Due to the data incompleteness issue of the collected time series, we set the element $x_{i,t}^k$ as 0 to keep equally spaced intervals for missing measurement.

Leveraging the Day-long Time Series, HeartSpace has the following three key steps: (i) day-specific time-series encoding to map each day-long time series into a representation vector; (ii) temporal pattern aggregation network to fuse day-specific representations; and (iii) Siamese-triplet network optimization strategy to retain the relations of implicit intra-series temporal consistency and inter-series discrepancy. The model training flow is shown in Figure 1.

3.1 Day-Specific Time-Series Encoding

To discover the underlying repeated local patterns and reduce dimensions of day-long time-series data, we first leverage a convolutional autoencoder module to map each individual series $x_{i,t}$ into a common low-dimensional latent space. In general, the encoder first takes a day-long time series as the input and then maps it into a latent representation which encodes the temporal pattern. Then, the decoder network reconstructs the data which is identical with the input $x_{i,t}$ in the ideal case.

To deal with missing values existed in time-series data, a straightforward solution is to impute missing values with zero for keeping the input dimension consistent with complete day-specific timeseries. However, directly applying zero padding in this scenario leads to a consequence, where the imputed values are treated the same as other valid inputs. They will both go through the kernels in each convolutional layer without distinction. It follows that the generated features get incorrectly encoded and these incorrect features will further get propagated from lower to higher layers. To resolve this issue stemming from zero padding, we introduce a dual-stage gating mechanism into the convolutional autoencoder module to re-weight the hidden units. We posit that would mitigate the undesired effects from zero padding. More specifically, each layer of our deep autoencoder framework is a four-step block: (i) convolutional network; (ii) channelwise gating mechanism; (iii) temporal gating mechanism; and (iv) pooling operation. Figure 2 presents the architecture of our deep autoencoder module.

3.1.1 Convolutional Layers. Firstly, we apply convolutional layers to encode the local pattern of day-long time series $x_{i,t}$. Specifically, we feed $x_{i,t}$ into a number of convolutional layers. Let’s denote $V_{i,t}^{l-1}$ as the feature map representation of the $(l-1)$-th layer. The output of the $l$-th layer is given as

$$V_{i,t}^l = f \left( W_c^l \ast V_{i,t}^{l-1} + b_c^l \right),$$

(1)
Representation Learning on Variable Length

Fig. 1. HeartSpace Framework. The training flow can be summarized as (1) feeding partition raw sensory data to a set of day-long time series; (2) mapping each day-long time series into a representation vector by Day-Specific Time-Series Encoding Module (see detailed structure in Figure 2); (3) fusing day-specific representations through Temporal Pattern Aggregation Network, where a temporal attention network is designed to capture the temporal relevance (see detailed structure in Figure 3); and (4) optimizing parameters based on reconstruction loss and Siamese-triplet loss, where the Siamese-triplet loss is constructed by sampling intra-series and inter-series.

where \( f(\cdot) \) is the activation function and \( \ast \) denotes the convolutional operation. \( \mathbf{W}_l^l \) and \( \mathbf{b}_l^l \) represent the transformation matrix and bias term in the \( l \)-th layer, respectively.

3.1.2 Channelwise Gating Mechanism. In our deep autoencoder module, the goal of the channelwise gating mechanism is to alleviate the side effect caused by zero padding. We propose to re-weight hidden units by exploiting the cross-channel dependencies and selecting the most informative elements from the encoded feature representation \( \mathbf{V}_{l,t}^l \) [15]. To exploit the dependencies over channel dimension, we first apply temporal average pooling operation \( \mathcal{F}_{\text{pool}}^{cg}(\cdot) \) on the feature representation \( \mathbf{V}_{l,t}^l \) over temporal dimension (1 ≤ \( k \) ≤ \( K \)) to produce the summary of each channelwise representation as

\[
\mathbf{z}_{l,t}^l = \mathcal{F}_{\text{pool}}^{cg} (\mathbf{V}_{l,t}^l) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{V}_{l,t,k,:}^l.
\] 

where \( \mathbf{z}_{l,t}^l \) is the intermediate representation of \( \mathbf{V}_{l,t}^l \) after average pooling operation over temporal dimension. Then, our channelwise gating mechanism recalibrates the information distribution.
among all elements across channels as

\[ a^l_{i,t} = \text{Sigmoid} \left( W^c_{2} \cdot \text{ReLU} \left( W^c_{1} \cdot z^l_{i,t} \right) \right), \]  

(3)

where \( a^l_{i,t} \) denotes the channelwise importance vector in which each entry is each channel’s importance. \( W^c_{1} \) and \( W^c_{2} \) are learned parameters of fully connected networks. Finally, the channelwise representation \( \tilde{V}^l_{i,t} \) is learned as

\[ \tilde{V}^l_{i,t,c} = V^l_{i,t,c} a^l_{i,t,c}, \]  

(4)

where \( c \) is the channel index.

3.1.3 Temporalwise Gating Mechanism. Similar to the channelwise gating mechanism, the gating mechanism on temporal dimension is to further learn focus points in the time-ordered internal feature representation \( \tilde{V}^l_{i,t} \) (output from the channelwise gating mechanism). Similar to the gating mechanism procedures encoded in Equations (2), (3), and (4), we first apply the channel average pooling operation on feature representation \( \tilde{V}^l_{i,t} \) over channel dimension and get \( \tilde{V}^\prime_{i,t} \) as the summarized feature representation which jointly preserve the channelwise and temporal dependencies.

Loss Function in Deep Autoencoder Module. In our HeartSpace framework, we propose to capture the temporal patterns of series data based on an encoder-decoder architecture. We elaborate the configuration details of our encoder-decoder framework in the Appendix section. We formally define the reconstruction loss function as follows:

\[ \mathcal{L}^{ae}_{i} = \sum_t \|m \odot (D(\mathcal{E}(x_{i,t})) - x_{i,t})\|_2^2, \]  

(5)

where \( m \in \mathbb{R}^K \) is a binary mask vector corresponding to each element in \( x_{i,t} \). In particular, \( m^k = 1 \) if \( x^k_{i,t} \neq 0 \) (i.e., has measurement) and \( m^k = 0 \) otherwise. \( \odot \) is the elementwise product operation, \( x_{i,t} \) is the input, and \( \mathcal{E}(\cdot) \) and \( D(\cdot) \) represent the encoder and decoder function, respectively.
3.2 Temporal Pattern Aggregation Network

While applying the autoencoder framework to map day-long time series into a low-dimensional vector, the fusion of day-specific temporal patterns presents a challenge. To address this challenge, we develop a temporal pattern aggregation network which promotes the collaboration of different day-specific temporal units for conclusive cross-time representations. Figure 3 shows the architecture of our temporal pattern aggregation network which consists of three major modules: (i) context-aware time embedding module; (ii) multi-head aggregation network; and (iii) temporal attention network.

3.2.1 Context-Aware Time Embedding Module. From the deep autoencoder module, given the time series $x_i$ of user $u_i$, we learn a set of date-ordered latent representations with size of $T$, i.e., $\mathcal{G}_i = \{g_{i,1}, \ldots, g_{i,T}\}$. In reality, different people may exhibit different wearable-sensory data distribution due to specific daily routines [30]. To incorporate the temporal contextual signals into our learning framework, we further augment our model with a time-aware embedding module, which utilizes the relative time difference between the last timestep and each previous one. For example, given a time series with three-date information \{2018-10-01, 2018-10-20, 2018-10-25\}, we generate a date duration vector as \{24, 5, 0\} (the day duration between 2018-10-01 and 2018-10-25 is 24 days). To avoid the occurrence of untrained long date duration in testing instances, we adopt a timing signal method [39] to represent a non-trainable date embedding. Formally, the vector $e_t$ of the $t$-th day is derived as

$$e_{t,2i} = \sin \left( \frac{t}{10000^{2i/d_e}} \right); e_{t,2i+1} = \cos \left( \frac{t}{10000^{2i/d_e}} \right),$$

where $d_e$ is the embedding dimension ($2i + 1$ and $2i$ are the odd and even index, respectively, in the embedding vector). New context-aware latent vector $h_{i,t} \in \mathbb{R}^{d_e}$ is generated by elementwise adding each day-specific feature representation $g_{i,t}$ and date embedding $e_i$, to incorporate the temporal contextual signals into the learned embeddings.

3.2.2 Multi-Head Aggregation Network. During the pattern fusion process, we develop a multi-head attention mechanism that is integrated with a pointwise feedforward neural network layer to automatically learn the quantitative relevance in different representation subspaces across all context-aware temporal patterns. Specifically, given the $i$-th time series, we feed all context-aware day-specific embeddings $H_i = \{h_{i,0}, \ldots, h_{i,T}\}$ into a multi-head attention mechanism. Here, $M$-heads attention conducts the cross-time fusion process for $Q$ subspaces.
Each \( q \)-th attention involves a separate self-attention learning among \( H_i \) as

\[
\tilde{H}_i^q = \text{softmax} \left( \frac{W_1^q \cdot H_i (W_2^q \cdot H_i)^T}{\sqrt{d_q}} \right) W_3^q \cdot H_i,
\]

(7)

where \( W_1^q, W_2^q, W_3^q \in \mathbb{R}^{d_q \times d_e} \) represent the learned parameters of the \( q \)-th head attention mechanism, and \( d_q \) is the embedding dimension of the \( q \)-th head attention, i.e., \( d_q = d_e/Q \). Then, we concatenate each learned embedding vector \( \tilde{H}_i^q \) from each \( q \)-th head attention, and further capture the cross-head correlations as follows:

\[
\tilde{H}_i = W_c \cdot \text{concat} \left( \tilde{H}_i^1, \ldots, \tilde{H}_i^Q \right).
\]

(8)

\( W_c \in \mathbb{R}^{d_e \times d_e} \) models the correlations among head-specific embeddings. Hence, we jointly embed multi-modal dependency units into the space with the fused \( \tilde{H}_i \) using the multi-head attention network. The advantage of our multi-head attention network lies in the exploration of feature modeling in different representation spaces [44]. Then, we further feed the fused embedding \( \tilde{H}_i \) into two fully connected layers, which is defined as follows:

\[
\tilde{H}_i^f = W_2^f \cdot \text{ReLU} \left( W_1^f \cdot \tilde{H}_i + b_1^f \right) + b_2^f,
\]

(9)

where \( W_1^f, W_2^f \) and \( b_1^f, b_2^f \) are the weight matrix and bias in the feedforward layer. In our HeartSpace framework, we perform multi-head attention mechanism twice.

3.2.3 Temporal Attention Network. To further summarize the temporal relevance, we develop a temporal attention network to learn importance weights across time. Formally, our temporal attention module can be represented as follows:

\[
\alpha_i = \text{softmax} \left( c \cdot \text{Tanh} \left( W^d \tilde{H}_i + b^d \right) \right), \quad \tilde{g}_i = \sum_t \alpha_{i,t} g_{i,t}.
\]

(10)

Output \( \tilde{H}_i^f \) is first fed into a one fully connected layer and then together with the context vector \( c \), it generates the importance weights \( \alpha_i \) through the softmax function. The aggregated embedding \( \tilde{g}_i \) is calculated as a weighted sum of day-specific embeddings based on these learned importance weights. For simplicity, we denote our temporal pattern aggregation network as \( \tilde{g}_i = A(\tilde{g}_i) \).

3.3 Siamese-Triplet Network Optimization Strategy

Our goal is to embed each individual wearable-sensory time series into low-dimensional spaces, in which every time series is represented as an embedding vector. We develop a Siamese-triplet network optimization strategy to jointly model the structural information of intra-series temporal consistency and inter-series discrepancy in our learning process. In particular, within a series of sensing data points (e.g., heart rate records), even though the measurements change over time, they do not change drastically within a short time period, since they belong to the same user [37]. Additionally, the heart rate measurements sampled from consecutive time intervals (e.g., days) of the same people may have a more similar distribution as compared to sampling from different people [5]. Motivated by these observations, the key idea of our Siamese-triplet network optimization framework is to learn representations with the constrain-making intra-series data point pairs closer to each other and inter-series data point pairs further apart. We first define the following terms to be used in our optimization strategy.

Definition 3 (Reference Set \( \mathcal{R}_i \)). We define \( \mathcal{R}_i \) to represent the sampled reference set of user \( u_i \). In particular, \( \mathcal{R}_i = \{ r_i^1, \ldots, r_i^{N_r} \} \), where \( N_r \) is the size of reference set corresponding to the \( N_r \)
Initialize all parameters; $p \in L$.

$q$ is the coefficient which controls the weight of Siamese-triplet and each query (i.e., while

We define $N_n$ aggregate support set $\{u \in x \}$ and positive set $\{p \}$ from the generated positive query $A$.

A sample negative set · is presented as $N$ and inconsistency among different users, we optimize representations that preserve $N_q u$ = $+$ $(u \in \{p \})$

and negative query set $P$ defined in Definition 2. Each entry in $P_n$ and $P_q$ is the sampled day-specific time series from user $u_i$. $L_s$.

Algorithm 1: The Model Inference of HeartSpace.

Input: User set $U$; reference set size $N_r$, positive query size $N_p$, negative query size $N_q$

1. Initialize all parameters;

2. while not converge do

3. sample a set of users from user set $U$;

4. foreach user $u_i$ do

5. sample support set $R_i$ and positive set $P_i$;

6. sample negative set $Q_i$;

7. feed each entry in $R_i$, $P_i$ and $Q_i$ into autoencoder to get daily representations and compute $L_{ae}^i$;

8. aggregate support set $R_i$ and derive $L_s^i$;

9. end

10. Update all parameters w.r.t. $L_{joint} = \sum_i L_{ae}^i + L_s^i$.

end

sampled day-specific time series of $x_i$, defined in Definition 2. Each entry in $R_i$ represents the $n_r$-th sampled time series from user $i$.

Definition 4 (Positive Query Set $P_i$). We define $P_i = \{p_{1}^{n}, \ldots, p_{N_p}^{n}\}$ to denote the positive query set of user with size of $N_p$. Specifically, every entry $p_{n}^{n}$ represents the $n_p$-th sampled day-specific time series from user $u_i$.

Definition 5 (Negative Query Set $Q_i$). $Q_i = \{q_{1}^{n}, \ldots, q_{N_q}^{n}\}$ is defined as the negative query set for user $u_i$, which is the sampled $N_q$ day-specific time series from other users except user $u_i$, i.e., $u_i \setminus \{i' \neq i\}$.

Based on the above definitions, given a specific user $u_i$, we first aggregate the elements from user $u_i$’s reference set $R_i$ as $\overline{r}_i = A(R_i)$, where $A(\cdot)$ is the aggregation function which represents the developed temporal pattern aggregation network. Then, we compute the cosine similarity of aggregated reference element $\overline{r}_i$ and each query (i.e., $p_{n}^{n}$ and $q_{n}^{n}$) from the generated positive query set $P_i$ and negative query set $Q_i$. Formally, the similarity estimation function $sim$ is presented as follows:

$$
sim(\overline{r}_i, p_{n}^{n}) = \overline{r}_i \cdot (E(p_{n}^{n}))^T; n_p \in [1, \ldots, N_p];
$$

$$
sim(\overline{r}_i, q_{n}^{n}) = \overline{r}_i \cdot (E(q_{n}^{n}))^T; n_q \in [1, \ldots, N_q],
$$

where $sim(\overline{r}_i, p_{n}^{n}) \in \mathbb{R}^1$ and $sim(\overline{r}_i, q_{n}^{n}) \in \mathbb{R}^1$. By capturing the temporal consistency of each individual user $u_i$ and inconsistency among different users, we optimize representations that preserve inherent relationships between each user’s reference set and query set, i.e., time-series embeddings from the same user are closer to each other, while embeddings from different users are more differentiated from each other. Therefore, we formally define our loss function as follows:

$$
L_s^i = \sum_{n_p} \sum_{n_q} \max (0, sim(\overline{r}_i, q_{n}^{n}) - sim(\overline{r}_i, p_{n}^{n}) + \gamma),
$$

where $\gamma$ is the margin between two similarities. The objective function of the joint model is defined as $L_{joint} = \sum_i L_{ae}^i + \lambda L_s^i$, where $\lambda$ is the coefficient which controls the weight of Siamese-triplet loss. The model parameters can be derived by minimizing the loss function. We use Adam optimizer to learn the parameters of HeartSpace. The model optimization process is summarized in Algorithm 1.
Unsupervised and Semi-Supervised Learning Scenarios. HeartSpace is a general representation learning model that is flexible for both unsupervised (without labeled instances) and semi-supervised (limited number of labeled instances) learning. In semi-supervised learning, given the labeled time series and its target value, we take the learned representation vector as the input of a single-layer perceptron with a combined loss function, i.e., integrate the joint objective function $L_{joint}$ with the loss function based on cross-entropy (categorical values) or MSE (quantitative values).

4 EVALUATION

We comprehensively evaluate HeartSpace on several inference and prediction tasks: user identification, personality prediction, demographic inference, and job performance prediction. Our longitudinal real-world data comes from two different studies leveraging two different sensors and different population groups, thus allowing us to carefully vet and validate our findings. To robustly evaluate the accuracy and generalization of embeddings learned by HeartSpace, within the context of the aforementioned prediction tasks, we aim to answer the following questions:

- **Q1**: How does HeartSpace perform compared with state-of-the-art representation learning methods for the wearable-sensory time series, as represented by heart rate?
- **Q2**: How is the performance of HeartSpace with respect to different training/testing time periods in user identification?
- **Q3**: How do different components of HeartSpace (i.e., deep autoencoder module, multi-head aggregation network, and Siamese-triplet network optimization strategy) contribute to the HeartSpace performance?
- **Q4**: How do the key hyperparameters (e.g., reference set size $S_r$ and embedding dimension $d_e$) affect HeartSpace performance?
- **Q5**: How does HeartSpace achieve better embeddings?

4.1 Experimental Settings

4.1.1 Data Description. We considered the heart rate time-series data from two research projects which collect heart rate data and other types of information (e.g., sleep and demographics) from participants. For simplicity, we shall notate them based on the sensors Garmin and Fitbit, respectively.

- **Garmin Heart Rate Data.** This dataset comes from an ongoing research study\(^1\) of workplace performance which measures the physiological states of employees in multiple companies. This dataset is collected from 578 participants (ages between 21 and 68) by Garmin band from March 2017 to August 2018. Each measurement is formatted as (user id, heart rate, timestamp).
- **Fitbit Heart Rate Data.** We collect this dataset from a research project\(^2\) at University of Notre Dame which aims to collect survey and wearable data from an initial cohort of 698 students (ages between 17 and 20) who enrolled in the Fall semester of 2015. This dataset is collected by Fitbit Charge during the 2015/2016 and 2016/2017 academic years.

4.1.2 Data Distribution. Figure 4 shows the distribution of wearable-sensory time series in terms of time-series length $J_i$ and completeness degree (i.e., the ratio of non-zero elements in day-specific time series $x^{d_i}$) on both Garmin and Fitbit heart rate data. As depicted in Figure 4(a) and (b), different datasets have different time-series distributions. Furthermore, Figure 4(c) and

\(^1\)https://tesserae.nd.edu/.
\(^2\)http://sites.nd.edu/nethealth/.
Fig. 4. Distribution of time-series length (a)–(b) and completeness degree (c)–(d).

(d) reveals that data incompleteness is ubiquitous, e.g., there exists more than 20% day-specific time series with data incompleteness <0.8, which poses a further challenge for HeartSpace. These observations are the main motivation to develop a temporal pattern aggregation network and a dual-stage gating mechanism for handling variable-length time series with incomplete data.

4.1.3 Methods for Comparison. To justify the effectiveness of HeartSpace for representation learning on wearable-sensory data, we compared it with the following contemporary representation learning methods:

– **Convolutional Autoencoder (CAE)** [4]: CAE is a representation learning framework by applying convolutional autoencoder to map the time-series patterns into latent embeddings.

– **Deep Sequence Representation (DSR)** [1]: DSR is a general-purpose encoder-decoder feature representation model on sequential data with two deep LSTMs: one to map input sequence to vector space and another to map vector to the output sequence.

– **Multi-Level Recurrent Neural Networks (MLR)** [7]: MLR is a multi-level feature learning model to extract low- and mid-level features from raw time-series data. RNNs with bidirectional-LSTM architectures are employed to learn temporal patterns.

– **Sequence Transformer Networks (STN)** [27]: STN is an end-to-end trainable method for learning structural information of clinical time-series data, to capture temporal and magnitude invariances.

– **Long Short-Term Memory Fully Convolutional Network (FCN)** [16]: FCN is designed for multivariate time-series classification model by proposing a squeeze-and-excitation block.

– **Wave2Vec** [43]: It is a sequence representation learning framework using a skip-gram model by considering sequential contextual signals. Specifically, it takes the one-by-one data points that surround the target point within a defined window, to feed into a neural network for appear probability prediction. The window size and number of negative samples is set as 1 and 4, respectively.

– **DeepHeart** [5]: DeepHeart is a deep learning approach which models the temporal pattern of heart rate time-series data with the integration of the convolutional and recurrent neural network architecture.

4.1.4 Evaluation Protocols. To measure the effectiveness of HeartSpace for representation learning on heart rate data in downstream applications, we conduct multiple tasks, including user identification, personality prediction, user demographic inference, job performance prediction, and sleep duration prediction (Figure 5). These tasks reflect inferring qualified self from quantified self. We summarize the details for five different tasks as follows:
— **User Identification**: In the user identification evaluation, each of the aforementioned methods learns a mapping function to encode each of the day-specific time series data into a low-dimensional representation vector. Then, given multiple day-specific time-series data from one user and an unknown day-specific time series, the task is to predict whether the unknown time series is collected from the same user of the multiple day-specific time series. Specifically, we first map day-specific time series to embedding vector by utilizing each of the aforementioned methods. Then we aggregate embedding vectors of multiple day-specific time series to generate the reference vector. We apply mean-pooling operation on baselines and temporal aggregation network on HeartSpace during the aggregation process. After that, we take the elementwise product between the reference vector and the embedding vector of the single day-specific time series as the input. Here, we adopt a logistic regression classifier to learn the model from the training data and generate the prediction. The True Positives and True Negatives are the heart rate series that are correctly identified for user $u_i$ or not, respectively, by the classifier. The False Positives and False Negatives are the heart rate series that are misclassified as belonging to user $u_i$ or not, respectively.

— **Personality Prediction**: We considered two personality attributes of the participants, namely, Conscientiousness and Agreeableness. We considered a classification task of a binary prediction on these attributes—high or low for each participant $u_i$ [23]. In this task, the goal was to evaluate whether the embeddings learned from the heart rate are effective predictors of the personality types. We use the Logistic Regression classifier to make the prediction with the embedding as the feature space. The True Positives and True Negatives are the participants that are correctly classified by the classification method as high- and low-level, respectively. The False Positives and False Negatives are the high- and low-level participants that are misclassified, respectively. We only use the Garmin data for this task, as we only survey personality attributes for the subjects in the pool.

— **User Demographic Inference**: We evaluate whether user demographics, such as gender and age, are predictable by using a similar experimental construct as in [11]. The age information is categorized into four categories (i.e., Young: from 18 to 24; Young-Adult: from 23 to 34; Middle-age: from 39 to 49; and Senior: from 49 to 100).

— **Job Performance Prediction**: We also evaluate the performance of predicting each participant’s job performance (Individual Task Performance), which is categorized into three types, i.e., good, neutral, and poor [9]. We only use the Garmin data for this task, as we only collect job performance information for the subjects in the Garmin pool.

— **Sleep Duration Estimation**: We further conduct the sleep duration estimation task to evaluate the quality of learned representations. We utilize the Linear Regression model which takes the embedding vectors of the target day-long time series as the input and predicts the sleep duration of that day as the output.
We evaluate user identification and personality prediction tasks in terms of $F1$-score, Accuracy, and AUC. In addition, the user demographic inference and job performance prediction tasks, given the multi-class scenario, are evaluated using Macro-$F1$ and Micro-$F1$. The sleep duration task is evaluated in terms of $MSE$ and $MAE$. In the evaluation of various tasks, we use the semi-supervised settings on the Personality Prediction, User Demographic Inference, and Job Performance Prediction evaluation tasks, since the label is collected over the individual user instead of day-specific labels. For the other two tasks (i.e., User Identification and Sleep Duration Estimation), we first perform the unsupervised learning to infer the model parameters and then generate day-specific representations based on the learned model. After that, a separate classifier/regression method is adopted to make predictions with the obtained embeddings.

4.1.5 Training/Test Data Split. We summarize the details of training/test data partition for different evaluation tasks as follows:

— For demographic inference, personality prediction, job performance prediction, and sleep duration estimation task, we use the entire heart rate data for learning time-series embeddings, and split the labels with 60%, 10%, and 30% for training, validation, and test, respectively.

— For the user identification task, we split the datasets in chronological order. We first use Garmin data from March to December in 2017 (10 months), to learn parameters of HeartSpace for time-series embedding in an unsupervised fashion. Then, we leverage the data from January 2018 to August 2018 to evaluate the user identification performance based on the generated embeddings from the learned model. Specifically, we perform the training/test process over the period of January 2018 to August 2018 with a sliding window of two months, i.e., (Jan → training, Feb → test); ; ; (Jul → training, Aug → test). The training month provides the labels to learn the classifier and the test month is used to evaluate the prediction accuracy. Irrespective of the training/test month combination, the model parameters are learned on the basis of 2017 data in an unsupervised fashion. This allows us to also consider the generalization of performance over time. Moreover, to ensure the fairness of performance comparison, the day-long time series of users shown in the test set are not visible in the training set using our partition strategy. The training/test partition method on Fitbit data is similar as the Garmin data.

4.2 Reproducibility

We implemented all the deep learning baselines and the proposed HeartSpace framework with Tensorflow. For the sake of fair comparison, all experiments are conducted across all participants in the testing data and the average performance is reported. Furthermore, the validation was run 10 times and the average performance numbers are reported. The code is available at the link below.

For the sake of reproducibility, we summarized the hyperparameter settings of HeartSpace in Table 1. In our experiments, we utilized the Glorot initialization [12] and grid search [14] strategies for hyperparameter initialization and tuning of all compared methods. The early stopping [32] is adopted to terminate the training process based on the validation performance. After the parameter tuning on all baselines, we reported their best performance in the evaluation results. During the model learning process, we used the Adam optimizer for all gradient-based methods, where the batch size and learning rate were set as 64 and 0.001, respectively.

The code is available at https://github.com/heartspace/heartspace.
### Table 1. Parameter Settings

| Parameter               | Value | Parameter               | Value |
|-------------------------|-------|-------------------------|-------|
| Embedding Dimension     | 64    | Support Size            | 6     |
| Positive Sampling Size  | 2     | Negative Sampling Size  | 4     |
| Margin Value            | 1     | Siamese-triplet weight  | 0.1   |
| Batch Size              | 64    | Learning Rate           | 0.001 |

### Table 2. User Identification Performance on Garmin and Fitbit Datasets

#### Garmin Heart Rate Data

| Method  | F1    | Acc. | AUC    | F1    | Acc. | AUC    | F1    | Acc. | AUC    | F1    | Acc. | AUC    |
|---------|-------|------|--------|-------|------|--------|-------|------|--------|-------|------|--------|
| CAE     | 0.658 | 0.650| 0.711  | 0.680 | 0.672| 0.737  | 0.660 | 0.669| 0.742  | 0.684 | 0.698| 0.770 |
| DSR     | 0.649 | 0.643| 0.692  | 0.685 | 0.670| 0.727  | 0.708 | 0.708| 0.784  | 0.694 | 0.696| 0.772 |
| MLR     | 0.652 | 0.637| 0.685  | 0.689 | 0.670| 0.731  | 0.707 | 0.702| 0.773  | 0.703 | 0.704| 0.776 |
| STN     | 0.616 | 0.572| 0.598  | 0.732 | 0.737| 0.816  | 0.750 | 0.758| 0.843  | 0.733 | 0.740| 0.817 |
| FCN     | 0.633 | 0.640| 0.668  | 0.657 | 0.642| 0.708  | 0.713 | 0.709| 0.788  | 0.725 | 0.716| 0.781 |
| Wave2Vec| 0.743 | 0.751| 0.832  | 0.796 | 0.803| 0.884  | 0.816 | 0.822| 0.901  | 0.824 | 0.830| 0.909 |
| DeepHeart| 0.731| 0.738| 0.812  | 0.753 | 0.757| 0.834  | 0.766 | 0.773| 0.855  | 0.750 | 0.756| 0.844 |

#### Fitbit Heart Rate Data

| Method  | F1    | Acc. | AUC    | F1    | Acc. | AUC    | F1    | Acc. | AUC    | F1    | Acc. | AUC    |
|---------|-------|------|--------|-------|------|--------|-------|------|--------|-------|------|--------|
| CAE     | 0.579 | 0.577| 0.602  | 0.596 | 0.594| 0.641  | 0.602 | 0.587| 0.634  | 0.606 | 0.598| 0.639 |
| DSR     | 0.649 | 0.646| 0.696  | 0.630 | 0.617| 0.654  | 0.619 | 0.612| 0.671  | 0.613 | 0.598| 0.641 |
| MLR     | 0.643 | 0.639| 0.697  | 0.626 | 0.624| 0.680  | 0.674 | 0.665| 0.740  | 0.659 | 0.649| 0.705 |
| STN     | 0.712 | 0.718| 0.778  | 0.687 | 0.689| 0.749  | 0.686 | 0.681| 0.746  | 0.729 | 0.727| 0.787 |
| FCN     | 0.703 | 0.695| 0.752  | 0.631 | 0.627| 0.675  | 0.693 | 0.690| 0.758  | 0.688 | 0.671| 0.762 |
| Wave2Vec| 0.778 | 0.778| 0.862  | 0.777 | 0.777| 0.857  | 0.771 | 0.770| 0.857  | 0.813 | 0.812| 0.893 |
| DeepHeart| 0.678| 0.680| 0.746  | 0.715 | 0.718| 0.787  | 0.719 | 0.722| 0.791  | 0.715 | 0.714| 0.784 |

Results of best baseline are underlined. Results of best baseline are underlined.

†(*) indicates that the result is significant according to Student’s t-test at level 0.01 (0.05) compared to the best baseline.

### 4.3 Performance Comparison (Q1 and Q2)

#### 4.3.1 User Identification

Table 2 shows the user identification performance on both datasets. We can observe that HeartSpace achieves the best performance and obtains significant improvement over state-of-the-art methods in all cases. This sheds light on the benefit of HeartSpace, which effectively captures the unique signatures of individuals. Although other neural network–based methods preserve the temporal structural information to learn latent representations for each individual time series, they ignore the variable length and incompleteness of wearable-sensory data, which reveals the practical difficulties in learning accurate models across non-continuous timesteps.

To make a thorough evaluation, we conduct a comparison experiment of HeartSpace and all baselines across different training and test time periods (e.g., Feb, Apr, Jun and Aug). We can note that the best performance is consistently achieved by HeartSpace with different forecasting time periods, which reflects the robustness of HeartSpace in learning the latent representations over time.

#### 4.3.2 Demographic Inference

The demographic inference performance comparison between HeartSpace and other competitive methods on the Garmin heart rate data is shown in Table 3. We can note that our HeartSpace outperforms other baselines in inferring users’ age and gender information, which further demonstrate the efficacy of our HeartSpace in learning significantly better...
Table 3. User Demographic Inference Results

| Demographic | Age  | Gender |
|-------------|------|--------|
|             | Micro-F1 | Macro-F1 | Micro-F1 | Macro-F1 |
| CAE         | 0.517  | 0.364  | 0.674  | 0.655  |
| DSR         | 0.505  | 0.259  | 0.645  | 0.633  |
| MLR         | 0.523  | 0.270  | 0.645  | 0.628  |
| STN         | 0.529  | 0.346  | 0.627  | 0.622  |
| FCN         | 0.531  | 0.296  | 0.643  | 0.624  |
| Wave2Vec    | 0.546  | 0.354  | 0.657  | 0.645  |
| DeepHeart   | 0.494  | 0.266  | 0.639  | 0.618  |
| HeartSpace  | **0.558†** | **0.377†** | **0.703†** | **0.672†** |

Results of best baseline are underlined.
†(*) indicates the result is significant according to Student’s t-test at level 0.01 compared to the best baseline.

Fig. 6. Personality detection results. The legend of (a) is the same as (b).

time-series embeddings than existing state-of-the-art methods. Similar results can be observed for the Fitbit heart rate data. In summary, the advantage of HeartSpace lies in its proper consideration of comprehensive temporal pattern fusion for time-series data.

4.3.3 Personality Prediction. Figure 6 shows the personality prediction results on two different categories (i.e., Agreeableness and Conscientiousness). We can observe that HeartSpace achieves the best performance in all personality cases. The performance is followed by DSR which extracts both multi-level temporal features during the representation learning process. This further verifies the utility of temporal pattern fusion in mapping time-series data into common latent space.

4.3.4 Job Performance Prediction. The results of job performance prediction are presented in Figure 7. In these figures, we can notice that HeartSpace achieves the best performance in terms of Macro-F1 and Micro-F1.

4.3.5 Sleep Duration Estimation. We further evaluate the proposed HeartSpace with the application of forecasting the sleep duration of people. The performance evaluation results (measured by MSE and MAE) are presented in Figure 8. We can observe that significant performance improvements are consistently obtained by HeartSpace over state-of-the-art baselines, which further validates the effectiveness of HeartSpace.

4.3.6 Discussion of Computational Costs. In this subsection, we discuss detailed computational costs during the training and testing periods. Specifically, the prediction time of (one sample) for
HeartSpace is (in seconds): 6.08×10^{-4}, whereas the most competitive baseline (Wave2vec) is 5.13×10^{-4}. The overall training times for HeartSpace and Wave2vec (in minutes) are 309 and 283.

4.4 Ablation Study: Componentwise Evaluation of HeartSpace (Q3)

We also aim to get a better understanding of key components of HeartSpace. In our evaluation, we consider three variants of the proposed method corresponding to different analytical aspects:

- **Effect of Siamese-Triplet Network.** HeartSpace-s. A simplified version of HeartSpace which does not include Siamese-triplet network to model intra- and inter-time series inter-dependencies.

- **Effect of Deep Autoencoder Module.** HeartSpace-a. A simplified version of HeartSpace without deep autoencoder module, i.e., only consider $L_s$ (Siamese-Triplet Network) in the loss function.

- **Effect of Multi-Head Attention Network.** HeartSpace-h: A variant of HeartSpace without the multi-head attention network to learn the weights of different day-specific temporal patterns.

We report the results in Figure 9. Notice that the full version of HeartSpace achieves the best performance in all cases, which suggests (i) the efficacy of the designed Siamese-triplet network optimization strategy for preserving structural information of implicit intra- and inter-time series correlations; (ii) the effectiveness of HeartSpace in capturing complex temporal dependencies across timesteps for variable-length sensor data; and (iii) the effectiveness of HeartSpace in exploring feature modeling in different representation spaces during our pattern fusion process. As such, it is necessary to build a joint framework to capture multi-dimensional correlations in wearable-sensory time-series representation learning.
4.5 Hyperparameter Studies (Q4)

To investigate the robustness of HeartSpace, we examined how the different choices of five key parameters affect the performance of HeartSpace. Figure 11 shows the evaluation results as a function of one selected parameter when fixing others. Overall, we observe that HeartSpace is not strictly sensitive to these parameters and is able to reach high performance under a cost-effective parameter choice, which demonstrates the robustness of HeartSpace.

Furthermore, we can observe that the increase of prediction performance saturates as the representation dimensionality increases. This is because at the beginning, a larger value of embedding dimension brings a stronger representation power for the recent framework, but the further increase of dimension size of latent representations might lead to the overfitting issue. In our experiments, we set the dimension size as 64 due to the consideration of the performance and computational cost. We can observe that both the positive query set size and negative query set size, as well as Siamese-triplet loss coefficient have a relatively low impact on the model performance.

4.6 Case Study: Analysis of Time-Series Representations (Q5)

We employ t-SNE [38] to further visualize the low-dimensional time-series representations learned by HeartSpace and one selected baseline (DeepHeart) on the Garmin heart rate dataset. Figure 10 shows the visualization of day-specific time-series embeddings from randomly selected 15 users. We can observe that the embeddings from the same user could be identified by our HeartSpace and cluster them closer than other embeddings, while the embeddings learned by DeepHeart of different users are mixed and cannot be well identified.
Fig. 11. Hyperparameter studies in terms of F1-score.

The qualitative results are further supported by the following quantitative analysis, where we calculate the averaged inner-group similarity for both DeepHeart and HeartSpace. Concretely, assume we have \( N \) groups, and each group includes \( M_i \) samples. The Averaged Inner-group Similarity (AIS) is defined as

\[
AIS = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{M_i (M_i - 1)} \sum_{j,k \in M_i} \| h_j - h_k \|_2,
\]

where \( h_j \) and \( h_k \) represent the embedding function of samples \( j \) and \( k \). The results of DeepHeart and HeartSpace are 123.78 and 35.91, where the superiority of HeartSpace demonstrates its promise in representing heart rate time series. Therefore, our model generates more accurate feature representations of user’s time-series data and is capable of preserving the unique signatures of each individual, which can then be leveraged for similarity analyses.

5 RELATED WORK

5.1 Representation Learning Models

With the advent of deep learning techniques, significant effort has been devoted to developing neural network–based representation learning models on various data. For the word representation learning, there exists a good amount of literature on designing techniques by considering word context in a document, such as word2vec [24] and Glove [29]. Many follow-up works extend the basic framework to learn latent representations on multimedia data [13, 31] and network data [14, 36], in order to capture frame-level temporal dependency and network structural information, respectively. For example, node2vec [14] proposes to learn node embeddings based on random walk. However, the wearable-sensory time-series representation learning remains as a critical but largely unsolved question. This work proposes a principled framework to address this challenge by automatically preserving the underlying structure of dynamic temporal patterns of wearable-sensory time series data.

5.2 Deep Learning for Healthcare Applications

With the advent of deep learning techniques, many deep neural network frameworks have been proposed to address various challenges in healthcare informatics [2, 6, 8, 21, 34]. For example, Cao et al. [6] proposed an end-to-end deep architecture for the prediction of human mood. Bai et al. [2] modeled sequential patient data and predict future clinical events. This work furthers this direction of investigation by proposing a general time-series representation learning framework to capture hierarchical structural correlations exhibited from human heart rate data, which cannot be handled by previous models. Ma et al. [20] employed an attentive bidirectional recurrent neural network to predict medical time-series data for healthcare.
5.3 Representation Learning on Wearable-Sensory Data

A handful of studies [1, 3, 17] have investigated the representation learning on wearable-sensory data. Ballinger et al. [3] proposed a semi-supervised learning method to detect health conditions. Amiriparian et al. [1] explored the deep feature representations to aid the diagnosis of cardiovascular disorders with sequence to sequence autoencoders. While these pioneering studies have demonstrated the effectiveness of representation learning on wearable-sensory data, they do not address our problem of designing a unified model that dynamically captures the evolving temporal characteristics from wearable-sensory time series with variable length.

6 CONCLUSION

In this article, we presented HeartSpace, a novel time-series representation learning method for wearable-sensory data that addresses several challenges stemming from such data and also overcomes the limitations of current state-of-the-art approaches, including dealing with incomplete and variable-length time series, intra-sensor/individual variability, and absence of ground truth. HeartSpace first learns latent representation to encode temporal patterns of individual day-specific time series with a deep autoencoder model. Then, an integrative framework of a pattern aggregation network and a Siamese-triplet network optimization maps variable-length wearable-sensory time series into the common latent space such that the implicit intra-series and inter-series correlations are well preserved. Extensive experiments on real-world data, representing different sensor and subjects’ distribution, demonstrate that the latent feature representations learned by HeartSpace are significantly more accurate and generalizable than the contemporary methods.

APPENDIX

In this section, we present the architecture configuration details of our deep autoencoder module.

Encoder. Given the day-long time series $x_{i,t} \in \mathbb{R}^K$, we use ReLU activation function with five convolutional layers (i.e., Conv1–Conv5) followed by channelwise, temporalwise gating mechanism and pooling layers. Particularly, Conv1–Conv5 are configured with the one-dimensional kernel with $\{9, 7, 7, 5, 5\}$ and filter sizes with $\{32, 64, 64, 128, 128\}$, respectively. Then, we perform the flatten operation on the output to generate a one-dimensional feature representation and feed it into a fully connected layer with Tanh activation function to generate the final latent representation $g_{i,t}$ corresponding to the time series $x_{i,t}$. Note that the number of layers, kernel sizes, and filter sizes are hyperparameters that may be configurable and could vary by specific sample interval (e.g., 30 s) of sensory data.

Decoder. The decoder is symmetric to the encoder in terms of the layer structure. First, the representation $g_{i,t}$ is uncompressed by a single layer with Tanh activation function and then followed by a series of five deconvolutional layers with ReLU activation function. The kernel and filter sizes are in reverse order to be symmetric to the encoder architecture configuration. Channelwise and Temporalwise are applied in the first four layers after deconvolutional operation.

REFERENCES

[1] Shahin Amiriparian, Maximilian Schmitt, Nicholas Cummins, Kun Qian, Fengquan Dong, and Björn Schuller. 2018. Deep unsupervised representation learning for abnormal heart sound classification. In EMBC. IEEE, 4776–4779.
[2] Tian Bai, Shanshan Zhang, Brian L. Egleston, and Slobodan Vucetic. 2018. Interpretable representation learning for healthcare via capturing disease progression through time. In KDD. ACM, 43–51.
[3] Xian Wu, Chao Huang, Pablo Robles Granda, and Nitesh V. Chawla. 2018. DeepHeart: Semi-supervised sequence learning for cardiovascular risk prediction. In IJCAI.
[23] Robert R. McCrae and Oliver P. John. 1992. An introduction to the five-factor model and its applications.
[22] Alireza Manashty, Janet Light, and Hamid Soleimani. 2018. A concise temporal data representation model for pre-
[21] Fenglong Ma, Jing Gao, et al. 2018. Risk prediction on electronic health records with prior medical knowledge. In
[15] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In KDD. ACM, 747–755.
[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In KDD. ACM, 3111–3119.
[13] Ross Goroshin, Joan Bruna, Jonathan Tompson, et al. 2015. Unsupervised learning of spatiotemporally coherent
metrics. In ICCV. 4086–4093.
[12] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In AISTATS. 249–256.
[11] Yuxiao Dong, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation.
[10] Patrick Doetsch, Pavel Golik, and Hermann Ney. 2017. A comprehensive study of batch construction strategies for
recurrent neural networks in MXNet. arXiv:1705.02414. https://arxiv.org/abs/1705.02414.
[9] James M. Conway. 1999. Distinguishing contextual performance from task performance for managerial jobs. Journal of Applied Psychology 84, 1 (1999), 3.
[8] Edward Choi, Mohammad Taha Bahadori, et al. 2017. GRAM: Graph-based attention model for healthcare representation learning. In KDD. ACM, 787–795.
[7] Maowei Cheng, Worku J. Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In KDD. ACM, 15–24.
[6] Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S. Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, and
Alex D. Leow. 2017. Deepmood: Modeling mobile phone typing dynamics for mood detection. In KDD. ACM, 747–755.
[5] Tanuka Bhattacharjee, Shreyasi Datta, Deepan Das, Anirban Dutta Choudhury, Arpan Pal, and Prasanta Kumar Ghosh. 2018. A heart rate driven Kalman filter for continuous arousal trend monitoring. In EMBC. IEEE, 3572–3577.
[4] Kevin Bascol, Rémi Emonet, et al. 2016. Unsupervised interpretable pattern discovery in time series using autoen-
coders. In SSPPR. Springer, 427–438.
[3] Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. 2014. Early stopping and non-parametric regression: An optimal
data-dependent stopping rule. JMLR 15, 1 (2014), 335–366.
[2] Rupesh K. Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In NIPS. 2377–2385.
[34] Yoshihiko Suhara, Yinzhan Xu, and Alex’Sandy’ Pentland. 2017. Deepmood: Forecasting depressed mood based on self-reported histories via recurrent neural networks. In WWW. ACM, 715–724.

[35] Yuxing Sun, Biao Leng, et al. 2015. A novel wavelet-SVM short-time passenger flow prediction in Beijing subway system. Neurocomputing 166 (2015), 109–121.

[36] Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. 2018. Deep recursive network embedding with regular equivalence. In KDD. ACM, 2357–2366.

[37] Arijit Ukil, Soma Bandyopadhyay, Chetanya Puri, and Arpan Pal. 2016. Heart-trend: An affordable heart condition monitoring system exploiting morphological pattern. In ICASSP. IEEE, 6260–6264.

[38] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of Machine Learning Research 9, 11 (2008).

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. 2017. Attention is all you need. In NIPS. 5998–6008.

[40] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In KDD. ACM, 2447–2456.

[41] Pengfei Wang, Jiafeng Guo, Yanyan Lan, and Jun Xu. 2016. Your cart tells you: Inferring demographic attributes from purchase data. In WSDM. ACM, 173–182.

[42] Ping Yang, Guy Dumont, et al. 2006. Adaptive change detection in heart rate trend monitoring in anesthetized children. IEEE Transactions on Biomedical Engineering 53, 11 (2006), 2211–2219.

[43] Ye Yuan, Guangxu Xun, et al. 2019. Wave2vec: Deep representation learning for clinical temporal data. Neurocomputing 324 (2019), 31–42.

[44] Jian Zhang, Xingjian Shi, Junyuan Xie, et al. 2018. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In UAI.

[45] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, et al. 2018. Matrix profile XI: SCRIMP++: Time series motif discovery at interactive speeds. In ICDM. IEEE, 837–846.

Received August 2020; revised December 2021; accepted April 2022