GRAPHSENSOR: A GRAPH ATTENTION NETWORK FOR TIME-SERIES SENSOR DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Our work focuses on the exploration of the internal relationships of signals in an individual sensor. In particular, we address the problem of not being able to evaluate such inter-sensor relationships due to missing rich and explicit feature representation. To solve this problem, we propose GRAPHSENSOR, a graph attention network, with a shared-weight convolution feature encoder to generate the signal segments and learn the internal relationships between them. Furthermore, we enrich the representation of the features by utilizing a multi-head approach when creating the internal relationship graph. Compared with traditional multi-head approaches, we propose a more efficient convolution-based multi-head mechanism, which only requires 56% of model parameters compared with the best multi-head baseline as demonstrated in the experiments. Moreover, GRAPHSENSOR is capable of achieving the state-of-the-art performance in the electroencephalography dataset and improving the accuracy by 13.8% compared to the best baseline in an inertial measurement unit (IMU) dataset.

1 INTRODUCTION

Recent years have seen a greater focus on human behavior recognition, especially monitoring health status of a person. The stress of modern living, coupled with poor living behaviors, has resulted in various chronic diseases, such as cervical spondylosis, endocrine dyscrasia, heart disease, and mental illnesses (Sharma & Majumdar, 2009). The sensor based approach to the detection of human behavior can assist in dealing with these problems. For instance, electroencephalography (EEG) may help diagnose a number of conditions including epilepsy (Siddiqui et al., 2013), sleep disorders, and brain tumors while surface electromyography (sEMG) and electrocardiography (ECG) can detect driver drowsiness (Lu et al., 2021). In addition, inertial measurement units (IMUs) are used to detect falls, heavy smoking, and alcohol intake (Rokni et al., 2018). Nowadays, since sensor fusion technology provides a more comprehensive view into people's behavior, graph based neural networks, such as Graph Convolutional Networks (GCNs) (Duvenaud et al., 2015) and Graph Attention Networks (GATs) (Veličković et al., 2018), have been developed for modeling sensor fusion data among various sensors.

However, it is also necessary to evaluate the internal relationships among time-series sensor data within an individual sensor, which was paid less attention in the previous work. Such internal relationships are crucial for human behavior recognition due to several reasons. Psychophysiological processes, for example, change across time due to the external and internal factors (Morales & Bowers, 2022), and therefore identifying what factors are relevant to the processes can measure dynamic changes and benefit the model inference (**Reason 1 - R1**) . Moreover, rich temporal information is embedded in the time-series sensor data (Morales & Bowers, 2022; Liu et al., 2021) (**Reason 2 - R2**). For instance, Alpha waves, Spindle waves, and K-complex waves can all be observed in sleep stages N1 and N2. However, in stage N1, Alpha waves predominate over Spindle waves and K-complex waves. On the other hand, as the sleep stage deepens, Spindle waves and K-complex waves become more prevalent than Alpha waves in stage N2 (Eldele et al., 2021). It shows some temporal correlations among such wave signals. We believe an in-depth internal relationship analysis among signal segments can improve the model inference with richer and more relevant features.

Deep learning techniques, such as convolutional neural networks (CNNs), have proven to be capable of learning useful representations from the underlying time-series data, but fail to elaborate on the

pair-wise connections (importance) between them. Using the standard multi-channel convolution method, channels are summed to form feature representations. Whereas using the depthwise separable convolution method, a single convolutional filter is applied to each channel, yet a pointwise convolution creates a linear combination (Chollet, 2017). Both of these prior works compromise the independence of the channels (e.g., salient waves). On the other hand, using graph attention neural networks can map each sensor as an independent node and learn the pairwise relationships between them. For an individual sensor, however, it is insufficient to construct a graph as the data from a single sensor has no explicit relationship to itself.

We propose GRAPHSENSOR to explore the internal relationships in an individual sensor, consisting of two key modules: (1) single segment representation module, and (2) relationship learning module. Inspired by the Vision Transformer (ViT), in the single segment representation module, we split individual sensor data into signal segments for evaluating the internal relationships. Through the application of an overlapping sliding window method, we aim to collect as many signal segments as possible from the raw data, thus preventing the loss of important signal patterns (information). The signal segments are then mapped to informative features. Instead of using the feature representation mechanism of CNN, which convolves across all signal segments, to maintain their independence, in this work, we apply a convolution encoder to every signal segment with shared weights. After that, all the signal segments are treated as an input to the relationship learning module. The relationship learning module consists of two blocks, *global node attention layers* and *graph based attention layers*. The former allows the model to selectively access what it considers to be the most important parts of the signal segments (**R1**), while the latter focus on segment-by-segment correlation analysis between these selected segments (**R2**). Moreover, we further enrich the feature representation by using a multi-head method when constructing the internal relationship graph. Compared with the standard multi-head methods, we propose a convolution-based multi-head approach, where multi-head features per signal segment are convolved to enrich the feature representation in a more efficient way. Comparatively, the proposed multi-head attention allows us to expand heads using lesser parameters and training time than the standard approach (more discussions can be found in Appendix C.5). In a nutshell, the main contributions of this paper are as follows:

- Our work is one of the first to explore inter-sensor relationships using convolution encoder for feature representation and graph-based self-attention for establishing antisymmetric correlations among signal segments.
- To improve the efficiency of our approach, we propose a convolution-based multi-head attention, which allows the heads to be expanded with less parameters and training time than the traditional multi-head approach.
- The experimental results demonstrate that our method can learn internal relationships between signal segments while achieving the state-of-the-art performance. Through the ablation studies across different settings, we provide an analysis of the effectiveness of the method's key components. The implementation and experimental details of GRAPHSENSOR are available at an anonymous site[1].

2 GRAPHSENSOR

Unlike the standard CNN, GRAPHSENSOR in segment representation (Section 2.1) encodes each signal segment individually. Moreover, in contrast to a regular RNN, GRAPHSENSOR in relationship learning (Section 2.2) assigns different attention weights to each signal segment. These attention weights designate how important or relevant a given signal segment is at a given time step. GRAPHSENSOR using graph-based self-attention learns the rich temporal relationships between the signal segments. In the graph-based self-attention, the edge between two signal segments is created only when the corresponding Pearson correlation coefficient is positive to rule out the irrelevant relation. Two attention coefficients are created for every pair of the signal segments to capture the temporal dependencies in Section 2.2.2. Multi-head method is applied to self attention. Our convolution-based multi-head self attention achieves a similar performance while requiring fewer parameters than the standard transformer (see discussion in Appendix C.5). We list the notations that are used in this work for convenience (as shown in Table 1).

[1] https://github.com/graphsensor/graphsensor-code

Table 1: Notations.

Symbol	Definition
L $(L \in \mathbb{N}, L > 1)$	Length of the time series
K $(K \in \mathbb{N}, K > 1)$	Number of the signal segments
D $(D \in \mathbb{N}, D > 1)$	Dimension of the signal segments
C $(C \in \mathbb{N}, C > 1)$	Dimension of the signal segment representation before $\mathcal{G}_{rl}$
E $(E \in \mathbb{N}, E > 1)$	Dimension of the signal segment representation after $\mathcal{G}_{rl}$

Overall Architecture: the proposed method (see Fig. 1) consists of two major parts, *signal segment representation* $\boldsymbol{\mathcal{F}_{sr}}(\mathcal{S}) = \mathcal{V}$ and *relationship learning* $\boldsymbol{\mathcal{G}_{rl}}(\mathcal{V}) = \mathcal{V}_{mh}$, where $\mathcal{S}$ is a signal segment set, $\mathcal{V}$ is a feature map, and $\mathcal{V}_{mh}$ represents multi-head features.

Given a time series $X = \{x_1, x_2, ..., x_l\} \in \mathbb{R}^{1\times L}$, an overlapping sliding window method is used to obtain a signal segment set $\mathcal{S} = \{s_1, s_2, ..., s_K\} \in \mathbb{R}^{K\times D}$ that is then mapped to $\mathcal{V} = \{v_1, v_2, ..., v_K\} \in \mathbb{R}^{K\times C}$ by a convolution-based feature encoder $\boldsymbol{\mathcal{F}_{sr}}$, (see the details in Section 2.1). This is followed by the construction of a graph using a graph attention neural network $\boldsymbol{\mathcal{G}_{rl}}$ to learn signal internal relationships (see the details in Section 2.2). After graph feature V^* is extracted, the classification is performed using a two-layer neural network consisting of GELU and Softmax, and the cross-entropy is used as a loss function when optimizing classification.

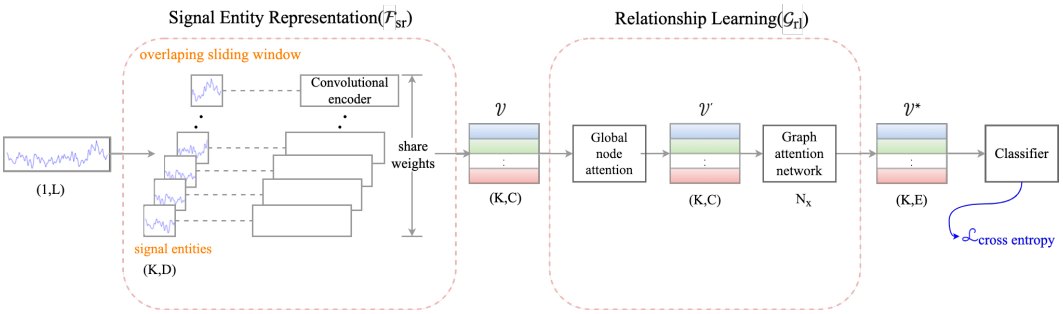

Figure 1: The overall architecture of GRAPHSENSOR.

2.1 Signal Segment Representation

Signal Segment Definition: Signal segments $\mathcal{S} \in \mathbb{R}^{K\times D}$ are used for measuring internal relationships within a given time series. An individual signal segment is defined as a stream of data values $s_i \in \mathbb{R}^{1\times D}$ at times t_i $(i \in \mathbb{N})$, where D is the length of a stream of data. Signal segments are generated from individual segments of a time series $X \in \mathbb{R}^{1\times L}$ that are split by a sliding window with a fixed length N $(N \in \mathbb{N}, N > 1)$ and an overlapping rate P $(P \in (0, 1))$. The length of the signal segment D is determined by the sliding window length, *i.e.*, $D = N$, while the number of the signal segments K is calculated as follows:

$$K = \left\lfloor \frac{L - N \times P}{N - N \times P} \right\rfloor \tag{1}$$

Signal Segment Representation: Signal segment representation is a convolutional encoder which maps signal segments $\mathcal{S} = \{s_1, s_2, ..., s_K\} \in \mathbb{R}^{K\times D}$ into a feature space $\mathcal{V} = \{v_1, v_2, ..., v_K\} \in \mathbb{R}^{K\times C}$. Three 1D convolution layers are used in the convolutional encoder, where a large-kernel with the size of 1×49 has been used in the first 1D convolution layer for the purpose of boosting CNN's performance by significantly enlarging its effective receptive field (Ding et al., 2022). Following that, a smaller-kernel with a size of 1×7 is used on the remaining 2 layers to further extract the feature map. At the end of the encoder, we utilize a residual Squeeze-and-Excitation (SE) block (Hu et al., 2018) to recalibrate the features learned from previous convolution layers. Note that each signal segment (row) shares **the same encoder**. The detailed convolutional encoder architecture is described in Appendix B.

2.2 RELATIONSHIP LEARNING

2.2.1 GLOBAL NODE ATTENTION

After the signal segment representation, a global node attention is implemented. As the human behavior changes across time due to the external and internal factors (**R1**), we design a global node attention to learn the important factor vector $W_G \in \mathbb{R}^{K\times 1}$, where an adaptive average pooling is operated on $\mathcal{V}$ to project its dimension from $\mathbb{R}^{K\times C}$ to $\mathbb{R}^{K\times 1}$. The output in turn is used as an input to a two-layer fully-connected network to learn the important factor vector W_G:

$$W_G = \sigma_1\big(\boldsymbol{W_2}(\sigma_2(\boldsymbol{W_1}(Pooling(\mathcal{V}))))\big) \tag{2}$$

The encodings of signal segments $\mathcal{V}^{'}$ are then weighted accordingly:

$$\mathcal{V}^{'} = W_G \times \mathcal{V} \tag{3}$$

The detailed global node attention architecture is described in Appendix C.1.

2.2.2 GRAPH-BASED SELF ATTENTION

Adjacency Matrix: A number of previous studies have used Pearson correlations to derive the adjacency matrix (Sun et al., 2014; Tijms et al., 2014; Xue et al., 2019). In this study, the adjacency matrix is constructed to represent the connection relationship between the signal segments v_i and v_j based on the Pearson's correlation coefficient, forming the edge between the signal segments. It is notable that the correlation coefficient between two signal segments determines only the edge between them. No matter what the correlation coefficient is, as long as it is positive, an edge is formed between two segments. Two examples of relationship visualization using Pearson correlations is given in Appendix D.

Attention Coefficients: The attention coefficient matrix is usually symmetric, however, most time series data in the real world are causal in nature (Dahlhaus & Eichler, 2003) and their relationship is therefore antisymmetric. Taking this characteristic into account, and inspired by (Veličković et al., 2018), our attention mechanism with softmax normalization is expressed as:

$$\alpha_{ij} = \frac{exp(Sigmoid(\mathbf{a}[\,pos_1 \times v_i \| pos_2 \times v_j]))}{\sum_{k\in\mathcal{N}_i} exp(Sigmoid(\mathbf{a}[\,pos_1 \times v_i \| pos_2 \times v_k]))} \tag{4}$$

where pos_1 and pos_2 are the positional encoding, $\|$ is the concatenation operation, and applying the Sigmoid nonlinearity. Here, **a** is a self-attention mechanism based on a two-layer fully-connected layer, where the signal segment representation matrix $\mathcal{V}$ is expanded from $\mathbb{R}^{K\times C}$ to $\mathbb{R}^{K\times K\times 2C}$ via a concatenation operation. After that, an adaptive average pooling is operated on the new signal segment representation matrix to project its dimension from $\mathbb{R}^{K\times K\times 2C}$ to $\mathbb{R}^{K\times K\times 1}$, which in turn is used as an input to a two-layer fully-connected network to learn the attention coefficient vector. By permuting the attention coefficient vector ($\in \mathbb{R}^{K^2}$), we obtain the attention coefficient matrix ($\in \mathbb{R}^{K\times K}$). The detailed graph-based self attention architecture is described in Appendix C.2.

Positional Encoding: The positional encoding in this study is derived from earlier positional encoding approaches in NLP, where the first word is given "1", the second word is given "2", and so on (Wang et al., 2019). Similarly, we present "1" for the first segment and "2" for the second segment.

2.2.3 MULTI-HEAD ATTENTION

In this work, the multi-head matrix is regarded as one feature map, and valuable features are extracted using a convolutional layer (as shown in Fig 2). The self-attention module in this work is split into two blocks: the *dense* layers and the *attention* layers (Pan et al., 2022).

Dense layers. Initially, 1×1 convolutions are conducted in the dense layer to project the signal segments to a higher dimension space (Veličković et al., 2018), where a multi-head method is applied to stabilize self-attention learning. As the multiple-head attention block is stacked many times in our work and in the first run, a dummy axis is used to expand the input tensor, which results in a change of input size from $\mathbb{R}^{K\times C}$ to $\mathbb{R}^{1\times K\times C}$ (i.e., $J = 1$ in Fig 2). The expanded dimension is used to generate the independent attention heads (the detailed architecture can be found in Appendix C.3).

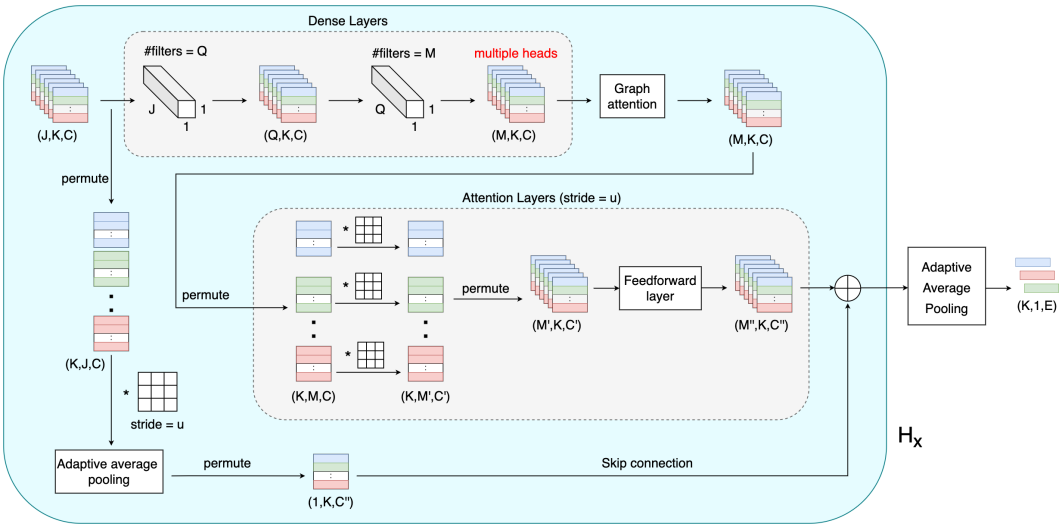

Figure 2: Illustration of the multi-head mechanism.

Attention Layers. To extract the signal segment features from $\boldsymbol{\mathcal{V}_{mh}}$, we permute the channels of the signal segments from the second dimension to the first dimension ($\mathbb{R}^{M \times K \times C} \rightarrow \mathbb{R}^{K \times M \times C}$). Feature extraction ($\mathbb{R}^{K \times M \times C} \rightarrow \mathbb{R}^{K \times M^{'} \times C^{'}}$) is then performed using a depthwise convolution with the $group\ size = K$ to preserve the identity of signal segments, where $M^{'}$ and $C^{'}$ are the dimensions after mapping, and the padding in the depthwise convolution is set to the half of the kernel size (as in Howard et al. (2019)).

A two-layer convolutional neural network based feedforward layer is then applied to map the features from $\mathbb{R}^{K \times M^{'} \times C^{'}}$ to $\mathbb{R}^{K \times M^{''} \times C^{''}}$. In addition, the input dimension is mapped to the same dimension of the output by applying a convolutional layer with the same stride and padding settings of the attention layers. This facilitates the stacking of skip connections. The multi-head attention is composed of a stack of H identical layers (see Appendix C.4). Eventually, the feature map of multiple heads is consolidated into a single head ($\mathbb{R}^{K \times 1 \times E}$) through an adaptive average pooling after the H identical layers.

3 Experiments

3.1 The Datasets and Implementation Details

Datasets: The experiments are conducted based on two widely used datasets, Sleep-EDF-20[2] (Eldele et al., 2021; Supratak et al., 2017; Sun et al., 2018; Phan et al., 2018; Mousavi et al., 2019) (SLEEP) and WISDM Smartphone and Smartwatch Activity and Biometrics Dataset[3] (Weiss et al., 2019; Mekruksavanich et al., 2020; Mekruksavanich & Jitpattanakul, 2020; Singh et al., 2020) (WISDM). The SLEEP dataset is used to assess the performance of the model on EEG signals, and WISDM is used to assess the performance of the model on inertial measurement unit (IMU) signals. The specification of signal segment definition in SLEEP and WISDM datasets is listed in Appendix A.

Training: We implement the model with PyTorch. All experiments are conducted using a single NVIDIA RTX 2080Ti GPU. We use the Adam optimizer for both datasets. The training consists of 100 epochs, starting at a learning rate of 0.001 for the first 10 epochs, then decaying to 0.0001 toward the end of the training. Random seed and batch size are set to 123 and 64, respectively.

[2]https://gist.github.com/emadeldeen24/a22691e36759934e53984289a94cb09b

[3]https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+

3.2 COMPARISON WITH PEER METHODS

We evaluate the benefits of our solution against a number of state-of-the-art methods (see the details in Appendix E), with the goal of understanding the following questions:

- **Q1: How does our model compare to the other baseline models?**
  Our model is compared to the other baseline models based on two publicly available datasets, SLEEP and WISDM. Specifically, in SLEEP the experiments are conducted by using the *Fpz-Cz* channel as the input, following the previous studies (Eldele et al., 2021; Supratak et al., 2017; Mousavi et al., 2019; Sun et al., 2018; Phan et al., 2018) used as the baselines. The datasets are prepared with 20-fold cross validation and all the 20 folds are used to train the model. When constructing and evaluating the internal relationships between the signal segments in an individual IMU signal from WISDM, the *accelerometer magnitudes* $= \sqrt{x^2 + y^2 + z^2}$ is calculated (as previous studies (Lu et al., 2020; Bhandari et al., 2017; Chow et al., 2021)) for each signal segment, where x, y, z are the three axes of an accelermometer. Most previous work has been developed to evaluate the multiple sensor data collected from sensors deployed simultaneously on different parts of the body, therefore these related models cannot be used to evaluate data collected from a single sensor. For comparison for WISDM, three state-of-the-art convolution based neural networks are applied as benchmarks, including EfficientNet b4 (Tan & Le, 2019), ResNet16 (Ismail Fawaz et al., 2020), and MobileNet V3 (Gupta, 2021), which have their input channels modified to 1 to allow them to fit the dataset. All the models are trained without using the pretrained weights.
- **Q2: How does our proposed multi-head attention solution compare to the state-of-the-art?**
  Our proposed multi-head attention solution is compared with the methods in standard Transformer (Veličković et al., 2018), and Graph Attention Networks (GAT) (Vaswani et al., 2017) using SLEEP. With the former, the output node feature dimension of a single head is reduced to ($\frac{d}{h}$) so as to increase the number of heads, where d is the total output node feature dimension and h is the number of head used. With the latter, the output node feature dimension of a single head remains unchanged to keep the original full dimensionality ($F^{'}$ defined on Page 3 in (Vaswani et al., 2017)).

Table 2: The performance comparison of the state-of-the-art approaches.

Datasets	Methods	Accuracy
SLEEP	DeepSleepNet (Supratak et al., 2017)	81.9
	ResnetLSTM (Sun et al., 2018)	82.5
	MultitaskCNN (Phan et al., 2018)	83.1
	SleepEEGNet (Mousavi et al., 2019)	81.5
	AttnSleepNet (Eldele et al., 2021)	84.4
	Ours	**84.3**
WISDM	EfficientNet b4 (Tan & Le, 2019)	27.0
	ResNet16 (Ismail Fawaz et al., 2020)	55.5
	MobileNet V3 (Gupta, 2021)	48.8
	Ours	**69.3**

We answer **Q1** using Table 2 which shows the comparison of our method with baseline algorithms on the SLEEP and the WISDM. Specifically, our method's accuracy is 84.3 on SLEEP. According to the results from SLEEP, our method with the adaptively learned graph representation is more suitable than many of the grid representations used by CNN and LSTM based algorithms (Supratak et al., 2017; Sun et al., 2018; Phan et al., 2018; Mousavi et al., 2019). Meanwhile, our method achieves the SOTA performance that is provided by AttnSleepNet (Eldele et al., 2021). However, it must be emphasized that AttnSleepNet includes two specially designed CNN modules, which cover detection of two specific bands of data: (1) using a filter with the size of 400 to extract the feature in 4-second windows for capturing the delta band, and (2) using a filter with the size of 50 to extract the feature in 0.5-second windows for capturing the alpha and theta bands, whereas our method covers

Table 3: The performance comparison of the multi-head approaches. (OM: GPU out of memory, #par: Model's number of parameters)

Methods	32 Heads(#par)	128 Heads(#par)	512 Heads (#par)
Transformer (Vaswani et al., 2017)	86.3 (2098K)	87.1 (3311K)	Error
GAT (Veličković et al., 2018)	85.4 (31897K)	86.4 (125416K)	OM (499494K)
Ours	**87.3 (1177K)**	**89.4 (1713K)**	**90.1 (7033K)**

general sensor data without a specific targeting band. The GRAPHSENSOR model also performs significantly better than other baseline models in WISDM target band, with an accuracy of 69.3.

We investigate the reason why GRAPHSENSOR performs much better in WISDM, and note that many activities in WISDM, such as walking, jogging, and stair climbing are confounding activities. In the traditional convolutional models, local features are extracted. In these confounding activities, these local features corresponding to a single arm movement are very similar. However, the relationships between these movements can differ. Whereas the traditional convolution-based models fail to detect such activities, GRAPHSENSOR recognizes the underlying relationship among signal segments (confounding movements), thus allowing for improved classification.

The response to **Q2** is given in Table 3, which shows how our performance compares to the baseline multi-head approaches using the SLEEP dataset. Our method achieves the best performance with the fewest parameters when the number of heads is 32 and 128. GAT-based multi-head approach (Veličković et al., 2018) runs out of GPU memory when the head number reaches 512, which indicates limited head scaling. After increasing the number of heads to 512, the transformation-based multi-head approach presents a run-time error. This occurs because the number of heads (h) is greater than the number of features ($d = 180$) of the output, resulting in $\frac{d}{h}$ approaching zero. In contrast, our method shows a gradual increase in performance with increasing head numbers, indicating a strong scalability. Moreover, the number of parameters used by GRAPHSENSOR is significantly much less than the best baselines ($1177K$ VS $2098K$ for 32 heads and $1713K$ VS $3311K$ for 128 heads). Such dramatic reduction of model parameters allow GRAPHSENSOR to be used for embedded devices for activity recognition and is significant contribution for the community. Investigating the applicability of using model compression (Cheng et al., 2017) on GRAPHSENSOR and conducting real-world experiments are our future work.

Table 4: Ablation and Sensitivity Study

Questions	Methods	Accuracy	#Parameters	Time Cost
Q3	GRAPHSENSOR (1,000, 5)	87.4	1253K	$\approx$ 1.0h
	GRAPHSENSOR (800, 6)	87.5	1323K	$\approx$ 1.0h
	GRAPHSENSOR (600, 9)	88.9	1545K	$\approx$ 1.7h
	GRAPHSENSOR (500, 11)	89.4	1713K	$\approx$ 1.7h
Q4	GRAPHSENSOR (500, 11) with head number 16	85.9	826K	$\approx$ 0.6h
	GRAPHSENSOR (500, 11) with head number 32	87.3	1177K	$\approx$ 0.6h
	GRAPHSENSOR (500, 11) with head number 128	89.4	1713K	$\approx$ 1.7h
	GRAPHSENSOR (500, 11) with head number 512	90.1	7033K	$\approx$ 3.0h

3.3 Ablation Study

To further investigate the effectiveness of our solution, we conduct an ablation study on the SLEEP dataset as shown in Table 4. Specifically, we derive a set of model variants from our solution with the goal of understanding the following questions:

- **Q3: How does the number of signal segments affect the performance of the solution?** GRAPHSENSOR *(x, y):* our model is investigated in terms of the number of signal segments. We vary the size of each signal segment x from $\{1000, 800, 600, 500\}$, corresponding to the number of signal segments y from $\{5, 6, 9, 11\}$.

- **Q4: How does the number of heads affect the performance of the solution?**
  Our model is investigated in terms of the the number of heads on GRAPHSENSOR (500, 11). We vary the number of heads from $\{16, 32, 128, 512\}$.
- **Q5: What are the effects of permuting the signal segments randomly on the model performance?**
- **Q6: How does GRAPHSENSOR perform when the original signals are already-short?**

For **Q3**, from Table 4, we can see that GRAPHSENSOR shows consistency in performance when given different numbers of signal segments. The results demonstrate that GRAPHSENSOR is able to stabilize the learning process of graphs even when the signal segment x varies greatly in size. On the other hand, we see that the model performance with similar numbers of signal segment is close, while using the higher number of signal segments (9 and 11) we can achieve a better result than using the smaller number of signal segments (5 and 6). The finding suggests that containing more signal segments in an individual sensor data can enrich its internal relationships, thus benefit the classification accuracy. However, a higher number of signal segments leads to a larger number of model parameters. The GRAPHSENSOR (500, 11) achieves the best accuracy, but has the most parameters. The time cost of a model is defined as its training speed in 100 epochs. The signal segments of size 9 and 11 have a higher accuracy, but they take longer to train than the signal segments of size 5 and 6. The results provide guidelines on how to balance accuracy and computation costs when using GRAPHSENSOR. We consider accuracy-first in this work and therefore use GRAPHSENSOR (500, 11) as the base setting for the classification problem.

The results of **Q4** in Table 4 provide a comparison of the performance of the model when varying the number of heads, indicating that the model performance increases from a smaller to a larger number of heads. Moreover, GRAPHSENSOR (500, 11) with head number 512 achieves highest performance (accuracy of 90.1). However, it requires about double of the training time and four times the parameters than the second-best setting (128 heads), while the improvement is only about 0.7%. These results help in understanding how attention heads can enhance learning on graphs at different costs. Considering the efficiency of GRAPHSENSOR (500,11), we use head number 128 in our experiment (Q1).

For **Q5**, we permute the signal segments with 500 data points (window size) randomly using the WISDM dataset. After permutation, the performance decreases from 69.3% (no permutation) to $68.9\%-67.8\%$ (5 random permutation rounds), demonstrating that GRAPHSENSO provides valuable information regarding temporal aspects.

For **Q6**, we perform a stress test on the WISDM dataset (as shown in Fig. 5), rearranging the length of the samples from $n = 200$ ($10s$) to $n = 20$ ($1s$). Our method still performs better than the other three state-of-the-art baselines even with the original length (EfficientNet -27%, ResNet - 55%, MobileNet V3 - 48.8%), despite the drop in accuracy from 69.3% to 56.1%. This suggests that our approach is practical to learn useful intra-sensor relationships even for a super-short sensor signal. The drop in accuracy is due to information loss.

Table 5: Stress test on the WISDM dataset.

Datasets	Methods	Accuracy	Sample Length (n)
WISDM	EfficientNet b4 (Tan & Le, 2019)	27.0	200
	ResNet16 (Ismail Fawaz et al., 2020)	55.5	200
	MobileNet V3 (Gupta, 2021)	48.8	200
	Ours	**56.1**	**20**

4 Related Work

Feature Representation: Convolutional Neural Networks (CNNs) have been proven to be very effective for feature representation. Each standard CNN model follows a similar design, where convolution is performed by sliding the filter over the input. An element-wise matrix multiplication is carried out at each location and the results are ultimately summed. This sum is then used to formulate

the feature map. Recently, CNN models, such as Restnet (He et al., 2016) and EfficientNet (Tan & Le, 2019), have become increasingly effective at object classification. In addition to the standard convolution based neural networks, Chollet (Chollet, 2017) proposed a new network structure called depthwise separable convolutional neural network, which includes depthwise and pointwise convolutions. As opposed to standard CNN's, where convolution occurs simultaneously for all channels, depthwise convolution is performed for a single channel at a time. A linear combination is then produced by applying pointwise convolution to all channels. One of the most popular models in this direction is MobileNet (Howard et al., 2019). However, the prior works compromise the independence of the channels and cannot be directly applied to map the signal segment features before learning the internal relationships. Another widely used approach for feature representation adopts Recurrent Neural Networks (RNNs). The models, such as Long Short-Time Memory (LSTM) (Greff et al., 2016) and Bi-directional Long Short-Term Memory (BiLSTM) (Graves et al., 2005), learn the important parts of the time-series data seen so far and forget the least important ones. While RNNs are used to learn a one-way relationship as a directed acyclic graph (Zhou et al., 2020), GRAPHSENSOR is able to learn the pairwise relationships between the single segments.

Relationship Learning: The non-Euclidean nature of the signal segment relationships makes it more appropriate to describe them with a graph data structure. Graph Neural Networks with attention mechanisms have demonstrated enhanced performance in dealing with graph structure data. Graph neural networks, such as the Graph Attention Network (GAT) (Veličković et al., 2018), compute attention scores using feed-forward neural networks, while the Graph convolutional neural Network (GCN) (Kipf & Welling, 2016) and GraphSAGE (Hamilton et al., 2017) computes attention scores by utilizing cosine similarity (dot product). Considering the fact that most time series data in the real world are causal in nature, and their relationship is asymmetric, the attention mechanism in GAT (Veličković et al., 2018) is more suitable for measuring the internal relationship between signal segments. We use GAT as one of our baselines. Another similar work can also be found in GraphSleepNet (Jia et al., 2020), which adaptively learns the external relationships of the time-series data. In addition, it uses multiple sensor nodes to enhance graph structure learning. Since our work explores one single sensor data, instead we look into the original multi-head attention (Vaswani et al., 2017). The attention mechanism is implemented by lowering the output dimension and then concatenating the respective outputs from the different heads. With lower dimensional outputs, more heads can be generated. The model appears to have more expressive power when it has more heads. Since many state-of-the-art works are built on top of this multi-head attention (Hamilton et al., 2017; Guo et al., 2020; Hao et al., 2020), we use the transformer architecture (Vaswani et al., 2017) as another baseline.

5 Conclusion

In this paper, we present GRAPHSENSOR, a novel deep graph attention network for modeling the internal relationships among signal segments generated by a single sensor, which does not receive enough attention yet from the research community. Specifically, GRAPHSENSOR is capable of generating graph nodes from the signal segments and efficiently learning the graph structure to determine the pairwise connections between the signal segments using convolution-based multi-head graph attention network. The experimental results demonstrate that GRAPHSENSOR achieves the state-of-the-art performance across two representative datasets and is more efficient than the baseline multi-head solutions (minimum 50% improvement in model size reduction) while having better accuracy. GRAPHSENSOR offers pioneering investigation into deep exploration of the interval relationships inside a single sensor's time-series data. We argue that such improvement is essential for sensor-based human behavior recognition. Moreover, the proposed multi-head mechanism is able to increase the number of heads to a great extent, which is appealing to explore opportunities that leverage GRAPHSENSOR for much diverse applications based on sensors. For instance, it is worthwhile to investigate how to apply GRAPHSENSOR to improve perception, prediction and planning models in autonomous driving relying on Lidar or multi-sensor fusion.

REFERENCES

Babin Bhandari, JianChao Lu, Xi Zheng, Sutharshan Rajasegarar, and Chandan Karmakar. Non-invasive sensor based automated smoking activity detection. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 845–848. IEEE, 2017.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

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

Daniel Hung Kay Chow, Luc Tremblay, Chor Yin Lam, Adrian Wai Yin Yeung, Wilson Ho Wu Cheng, and Peter Tin Wah Tse. Comparison between accelerometer and gyroscope in predicting level-ground running kinematics by treadmill running kinematics using a single wearable sensor. *Sensors*, 21(14):4633, 2021.

Rainer Dahlhaus and Michael Eichler. Causality and graphical models in time series analysis. *Oxford Statistical Science Series*, pp. 115–137, 2003.

Xiaohan Ding, Xiangyu Zhang, Yizhuang Zhou, Jungong Han, Guiguang Ding, and Jian Sun. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. *arXiv preprint arXiv:2203.06717*, 2022.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Emadeldeen Eldele, Zhenghua Chen, Chengyu Liu, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:809–818, 2021.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International conference on artificial neural networks*, pp. 799–804. Springer, 2005.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Xiangyang Xue, and Zheng Zhang. Multi-scale self-attention for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7847–7854, 2020.

Saurabh Gupta. Deep learning based human activity recognition (har) using wearable sensor data. *International Journal of Information Management Data Insights*, 1(2):100046, 2021.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. *arXiv preprint arXiv:2004.11207*, 2, 2020.

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

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

Ziyu Jia, Youfang Lin, Jing Wang, Ronghao Zhou, Xiaojun Ning, Yuanlai He, and Yaoshuai Zhao. Graphsleepnet: Adaptive spatial-temporal graph convolutional networks for sleep stage classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1324–1330, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Xun Liu, Danfeng Hong, Jocelyn Chanussot, Baojun Zhao, and Pedram Ghamisi. Modality translation in remote sensing time series. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2021.

Jianchao Lu, Xi Zheng, Michael Sheng, Jiong Jin, and Shui Yu. Efficient human activity recognition using a single wearable sensor. *IEEE Internet of Things Journal*, 7(11):11137–11146, 2020.

Jianchao Lu, Xi Zheng, Lihong Tang, Tianyi Zhang, Quan Z Sheng, Chen Wang, Jiong Jin, Shui Yu, and Wanlei Zhou. Can steering wheel detect your driving fatigue? *IEEE Transactions on Vehicular Technology*, 70(6):5537–5550, 2021.

Sakorn Mekruksavanich and Anuchit Jitpattanakul. Smartwatch-based human activity recognition using hybrid lstm network. In *2020 IEEE SENSORS*, pp. 1–4. IEEE, 2020.

Sakorn Mekruksavanich, Anuchit Jitpattanakul, Phichai Youplao, and Preecha Yupapin. Enhanced hand-oriented activity recognition based on smartwatch sensor data using lstms. *Symmetry*, 12(9): 1570, 2020.

Santiago Morales and Maureen E Bowers. Time-frequency analysis methods and their application in developmental eeg data. *Developmental Cognitive Neuroscience*, 54:101067, 2022.

Sajad Mousavi, Fatemeh Afghah, and U Rajendra Acharya. SleepEEGNet: Automated sleep stage scoring with sequence to sequence deep learning approach. *PloS One*, 14(5):e0216456, 2019.

Xuran Pan, Chunjiang Ge, Rui Lu, Shiji Song, Guanfu Chen, Zeyi Huang, and Gao Huang. On the integration of self-attention and convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 815–825, 2022.

Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y Chén, and Maarten De Vos. Joint classification and prediction cnn framework for automatic sleep stage classification. *IEEE Transactions on Biomedical Engineering*, 66(5):1285–1296, 2018.

Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. Personalized human activity recognition using convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Mukesh Sharma and PK Majumdar. Occupational lifestyle diseases: An emerging issue. *Indian journal of occupational and environmental medicine*, 13(3):109, 2009.

Mohd Maroof Siddiqui, Saifur Rahman, Syed Hasan Saeed, and Anurag Banodia. Eeg signals play major role to diagnose sleep disorder. *International Journal of Electronics and Computer Science Engineering (IJECSE)*, 2(2):503–505, 2013.

Satya P Singh, Madan Kumar Sharma, Aimé Lay-Ekuakille, Deepak Gangwar, and Sukrit Gupta. Deep convlstm with self-attention for human activity decoding using wearable sensors. *IEEE Sensors Journal*, 21(6):8575–8582, 2020.

Yu Sun, Qihua Yin, Rong Fang, Xiaoxiao Yan, Ying Wang, Anastasios Bezerianos, Huidong Tang, Fei Miao, and Junfeng Sun. Disrupted functional brain connectivity and its association to structural connectivity in amnestic mild cognitive impairment and alzheimer's disease. *PloS one*, 9(5): e96505, 2014.

Yudong Sun, Bei Wang, Jing Jin, and Xingyu Wang. Deep convolutional network method for automatic sleep stage classification based on neurophysiological signals. In *Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*, pp. 1–5, 2018. doi: 10.1109/CISP-BMEI.2018.8633058.

Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, 2017.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Betty M Tijms, Hiu M Yeung, Sietske AM Sikkes, Christiane Möller, Lieke L Smits, Cornelis J Stam, Philip Scheltens, Wiesje M van der Flier, and Frederik Barkhof. Single-subject gray matter graph properties and their relationship with cognitive impairment in early-and late-onset alzheimer's disease. *Brain Connectivity*, 4(5):337–346, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *Proceedings of the International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*, 2019.

Gary M Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202, 2019.

Hansheng Xue, Jiajie Peng, and Xuequn Shang. Towards gene function prediction via multi-networks representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 10069–10070, 2019.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

APPENDIX

A SPECIFICATION FOR SIGNAL SEGMENT DEFINITION

Table 6 lists the specification of signal segment definition in SLEEP and WISDM datasets.

Table 6: Specification for Signal Segment Definition.

Dataset	L	K	D	P
SLEEP	3,000	11	500	0.5
WISDM	200	9	40	0.5

In the SLEEP dataset, each Fpz-Cz channel has a length of $3,000$ data points (L), from which 11 signal segments (K) are generated using a sliding window method with the window size of 500 data points (D) and 50% overlapping rate (P). In the WISDM dataset, each accelerometer magnitude has a length of 200 data points (L), from which 9 signal segments (K) are generated using a sliding window method with the window size of 40 data points (D) and 50% overlapping rate (P).

B SIGNAL SEGMENT REPRESENTATION - CONVOLUTIONAL ENCODER

The convolutional encoder, as shown in Figure 3, starts from a 1D convolution layer with the input $X = \{x_1, x_2, ..., x_D\}, x_i \in \mathbb{R}$, where D is a hyperparameter that represents the length of a signal segment. After that, a max pooling is used to down sample the feature map from $I = Conv1D\_1(X), I \in \mathbb{R}^{A \times D}$ to $I' = Maxpooling\_1(I), I' \in \mathbb{R}^{A' \times D'}$, where A and A' denote the 1D convolution output size before and after the max pooling, D and D' refer to a length of feature sequence before and after the max pooling. Following this, two 1D convolution layers and another max pooling are sequentially implemented to I' such that $\varphi = Maxpooling\_2\big(Conv1D\_3(Conv1D\_2(I'))\big), \varphi \in \mathbb{R}^{A'' \times D''}$, where A'' is the output size after two 1D convolution layers and max pooling, and D'' is the

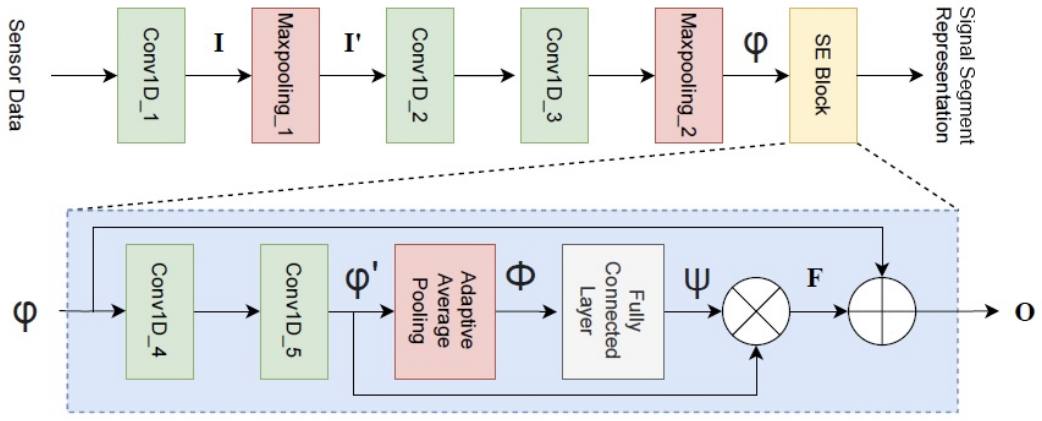

Figure 3: The Convolutional Encoder in Signal Segment Representation.

dimension of the feature output. Afterwards, we utilize a residual Squeeze-and-Excitation (SE) block (Hu et al., 2018) to recalibrate the features learned from previous convolutional layers, thus improving the performance. Given the feature map φ learned from previous convolutional layers, we apply two 1D convolution layers to $\boldsymbol{\varphi}$ such that $\boldsymbol{\varphi'} = Conv1D\_5(Conv1D\_4(\varphi)), \varphi' \in \mathbb{R}^{B\times M}$, where B is the output dimension, M is the length of features. A method of adaptive average pooling is then used to compress spatial information, thereby shrinking φ' to $\phi = Avgpooling(\varphi'), \phi \in \mathbb{R}^{B\times M}$. Next, two fully connected (FC) layers are applied to make use of the aggregated information $\psi = \delta(W_2\sigma(W_1(\phi))), \psi \in \mathbb{R}^{B\times M}$, where δ and σ refer to Sigmoid and GELU activation functions respectively, and W_1 and W_2 represent two FC layers. The feature map φ' is subsequently scaled by ψ as follows:

$$F = \boldsymbol{\varphi'} \otimes \psi \tag{5}$$

where $\otimes$ represents a point-wise multiplication between φ' and ψ. Lastly, a shortcut connection is applied to combine φ learned from previous convolutional layers with the enhanced features F derived from the residual SE block:

$$O = \sigma(\boldsymbol{\varphi} + F) \tag{6}$$

where σ is a GELU activation function. The output O is the spatial features extracted from the basis segments. Table 7 presents the parameters' value for the convolutional encoder.

Table 7: Specification for Convolutional Encoder.

Layer	Input channel	Output channel	Kernel size	Stride	Padding
$Conv1d\_1$	1	64	49	6	49//2
$Maxpooling\_1$	-	-	7	4	7//2
$Conv1d\_2$	64	128	7	1	7//2
$Conv1d\_3$	128	128	7	1	7//2
$Maxpooling\_2$	-	-	3	4	3//2
$Conv1d\_4$	128	30	1	1	0
$Conv1d\_5$	30	30	1	1	0

C Relationship Learning

C.1 Global Node Attention

The global node attention W_G is defined as:

$$W_G = \sigma_1\big(\boldsymbol{W_2}(\sigma_2(\boldsymbol{W_1}(Pooling(\mathcal{V}))))\big) \tag{7}$$

where W_1 and W_2 denote a double layer MLP, σ_2 is a GELU activation function, and σ_1 is a Sigmoid activation function.

As shown in Fig. 4, an adaptive average pooling is first used to squeeze the signal segment representation $\mathcal{V}$ from $\mathbb{R}^{K\times C}$ to $\mathbb{R}^{K\times 1}$. After that, two 1×1 convolution based FC layers are used to learn the attention score, where FC1 is a dimensionality-reduction layer with reduction ratio $r = 2$ to reduce the output of adaptive average pooling from $\mathbb{R}^{K\times 1}$ to $\mathbb{R}^{\frac{K}{r}\times 1}$. A nonlinear transformation (GELU) is then applied to

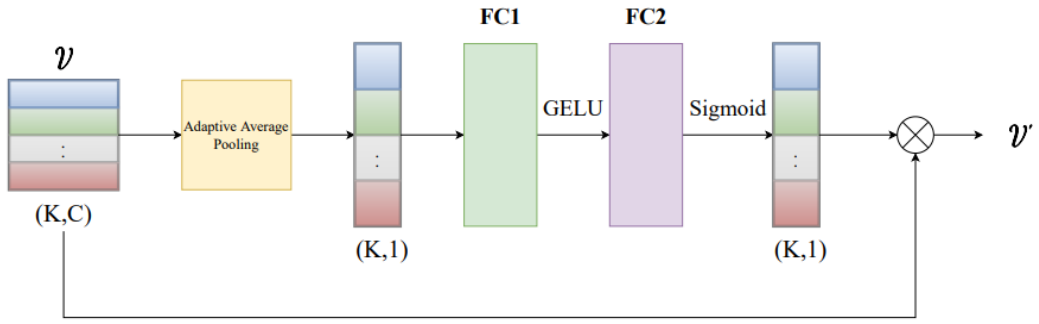

Figure 4: The architecture of the global node attention module.

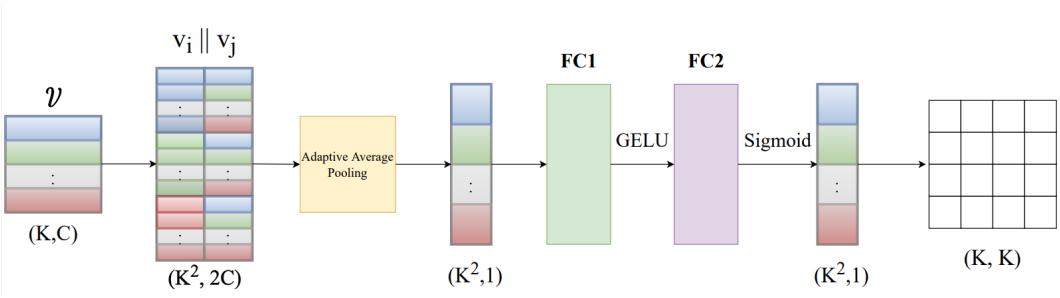

Figure 5: The architecture of graph attention.

FC1 output, followed by a dimensional-increasing layer FC2, which increases the channel dimension of FC1's output back to $\mathbb{R}^{K\times 1}$. Finally, we calculate the attention score by mapping the FC2's output between 0 and 1 using a Sigmoid function. The signal segment representation $\mathcal{V}'$ is updated using the $Eq.\ 2$ on Page 4 of the paper.

C.2 Graph Attention

The graph attention module has a similar architecture with the global node attention model as shown in Fig 5. An concatenation with positional encoding is firstly used to transform the signal segment representation $\mathcal{V}$ from $\mathbb{R}^{K\times C}$ to $\mathbb{R}^{K^2\times 2C}$. After that, an adaptive average pooling squeezes the concatenated signal segment representation from $\mathbb{R}^{K^2\times 2C}$ to $\mathbb{R}^{K^2\times 1}$. Then, two 1x1 convolution based FC layers with a GELU and Sigmoid activation functions are applied to learn the attention score, where FC1 reduces the concatenated signal segment representation from $\mathbb{R}^{K^2\times 1}$ to $\mathbb{R}^{\frac{K^2}{r}\times 1}$ with the reduction ratio $r = 2$, and FC2 maps it back to $\mathbb{R}^{K^2\times 1}$. After using the Sigmoid function to process the output, the result is reshaped into $\mathbb{R}^{K\times K}$ to represent the attention coefficients. Following the attention coefficients normalization, node aggregation is performed on the edges with exactly the direct neighbors of the central node and the central node itself:

$$v_i^* = \sum_{j\in\mathcal{N}_i} \alpha_{ij} v_j \tag{8}$$

where $\mathcal{N}_i$ is the neighborhood of node i in the graph, α_{ij} is the attention coefficient.

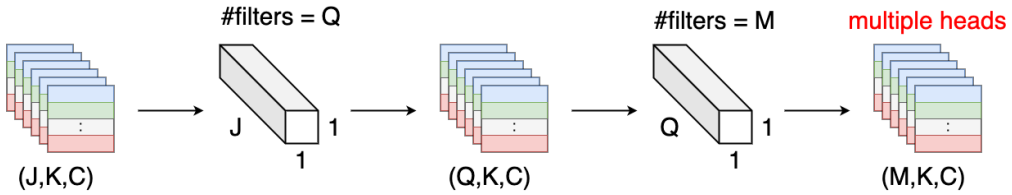

Figure 6: The architecture of dense layers.

C.3 DENSE LAYERS

Why Double-layer structure:

We discuss the model in the paper when $H = 1$, and now we move on to discussing the model when $H > 1$, where H represents the number of the graph attention network that will be stacked (as shown in Fig. 1).

- Layer 1: This layer summarizes information extracted by the previous attention layer when H >1. When $\#filters = 1$, a single set of summarized information is generated using a $1 \times 1 \times J$ filter, whereas diverse information with different focuses is generated when $\#filters > 1$.
- Layer 2: Using Layer 2, the summaries of the information obtained by Layer 1 are combined linearly to generate various multiple heads using a $1 \times 1 \times Q$ filter. When $\#filters = 1$, a new head is prepared for the graph attention implementation, while more heads are generated to benefit the graph learning process when $\#filters > 1$.

C.4 MULTI-HEAD ATTENTION

In the paper, the multi-head attention is composed of a stack of identical blocks ($H = 4$). The specification for the multi-head attention is illustrated in Table 8:

Table 8: Specification for multi-head attention ($H = 4$).

Layer	Input channel	Output channel (multi-head)	Kernel size	Stride	Padding
Block_1	1	64	5	2	5//2
Block_2	64	32	5	2	5//2
Block_3	32	64	5	1	5//2
Block_4	64	128	5	1	5//2

C.5 DISCUSSION

Based on the prior work (Vaswani et al., 2017), the standard multi-head attention concatenates every single-head from different representation subspaces at different positions, resulting in the final values using a fully-connected based feedforward layer as shown in Eq. 9:

$$MultiHead(standard) = Concat(head_1, ..., head_H)W^O \tag{9}$$

where W^O is a fully-connected based feedforward layer and H is the number of heads. Our multi-head attention mechanism stacks every single head together to

create a multi-channel feature map, where each channel represents an individual signal head. We then apply a convolution based module to process the feature map as shown in Eq. 10:

$$MultiHead(ours) = Stack(head_1, ..., head_H) \otimes K \tag{10}$$

where $\otimes$ is a convolution process, and K is a convolution filter (kernel).

In general, a convolutional layer is much more specialized, and efficient, than a fully connected layer. Each neuron in a fully connected layer is connected to every other neuron from the previous layer, and each connection has its own weight, so it is quite expensive to operate in terms of memory (weights) and computation (connections). Conversely, in a convolutional layer, each neuron is only connected to a few nearby neurons from the previous layer, and the weights (and local connections) are the same for all neurons. Due to the lower number of parameters used, convolution-based multi-head attention is more efficient than the standard one in terms of the reduction of model size (parameters) and training time as shown in Table 3 in the paper.

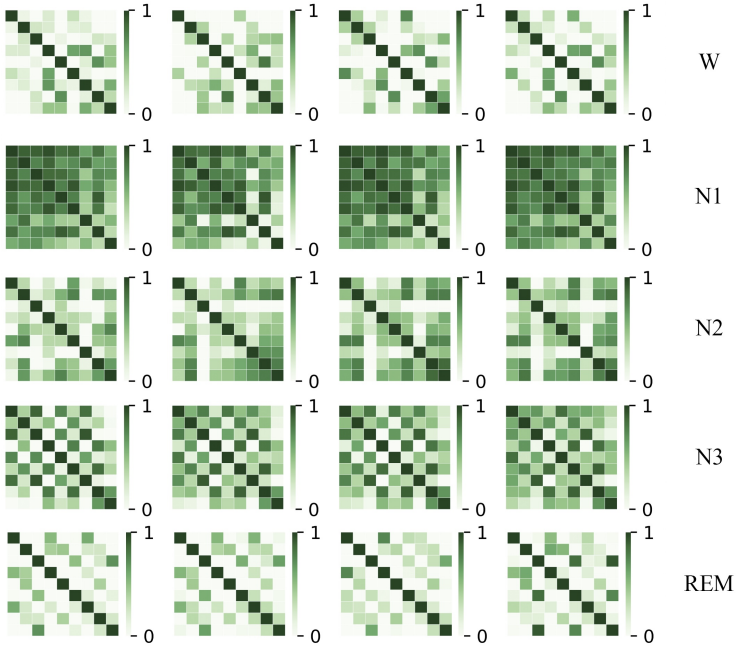

Figure 7: Illustration of the internal relationship using Pearson Correlation (SLEEP).

D RELATIONSHIP VISUALIZATION

SLEEP. There are five different stages of sleep (W, N1, N2, N3, REM) in SLEEP. The heatmaps as shown in Fig. 7 depict the internal relationship between the signal segments in EEG at different stages of sleep. Each row reflects the heatmaps for a specific stage of sleep and each heatmap on the row is retrieved from different batch representing different signal segments for the same sleep stage. As shown in Fig. 7, in each label, all the signal segments exhibit similar heatmap patterns but with distinct difference with signal segments from different labels (even from the

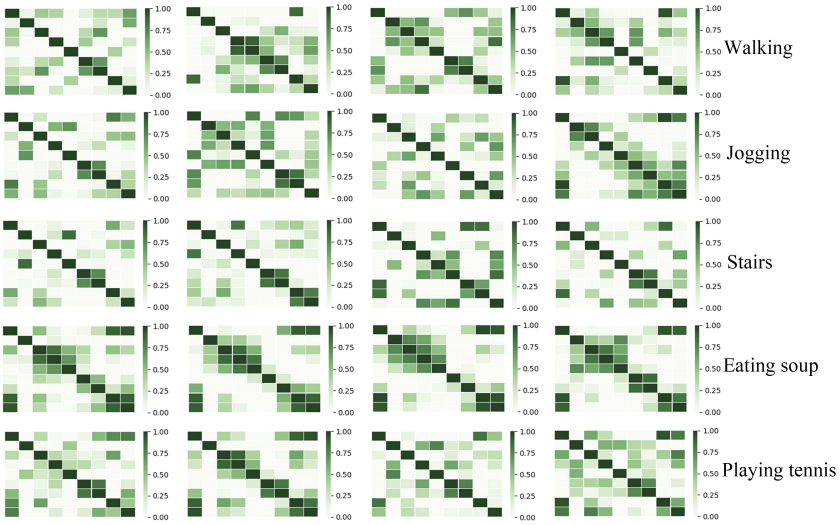

Figure 8: The internal relationship using Pearson Correlation (WISDM).

same column, that is from the same batch). This shows that GRAPHSENSOR indeed can learn unique relationship among signal segments for the same activity and human behavior can be classified with much better accuracy with such higher granularity feature representation.

WISDM. There are 18 different activities in the WISDM dataset. The heatmaps as shown in Fig. 8 depict the internal relationship between five activities (Walking, Jogging, Climbing Stairs, Eating Soup, and Playing Tennis Ball) using Pearson Correlation. From Fig. 8, we can observe that eating soup and playing tennis have unique patterns, while walking, jogging, and climbing stairs, as three confounding activities, have similar intra-sensor relationships.

E BASELINE METHODS IN SLEEP

- **DeepSleepNet (Supratak et al., 2017):** The model exploits a custom CNN architecture followed by an LSTM with a residual connection for sleep stage classification.
- **ResnetLSTM (Sun et al., 2018):** The model implements a ResNet architecture for feature extraction, followed by an LSTM to classify EEG signals into different sleep stages.
- **MultitaskCNN (Phan et al., 2018):** The model starts by converting the raw EEG signals into power spectrum images, and then applies a joint classification and prediction technique using a multi-task CNN architecture for identifying sleep stages.
- **SleepEEGNet (Mousavi et al., 2019):** This model employs the same CNN architecture as DeepSleepNet (Supratak et al., 2017) followed by an encoder-decoder with attention mechanism.
- **AttnSleepNet (Eldele et al., 2021):** The model starts with a feature extraction module that draws on a multi-resolution convolutional neural network

and adaptive feature recalibration, followed by a temporal context encoder that uses a multi-head attention mechanism to capture the temporal dependency feature for recognizing sleep stages.