

Article

GRAPHSENSOR: A Graph Attention Network for Time-Series Sensor

Jiaqi Ge ¹, Gaochao Xu ¹, Jianchao Lu ², Xu Xu ¹ and Xiangyu Meng ^{1,*}

¹ College of Computer Science and Technology, Jilin University, Changchun 130012, China; gejq18@mails.jlu.edu.cn (J.G.); xugc@jlu.edu.cn (G.X.); xuxu20@mails.jlu.edu.cn (X.X.)

² School of Computing, Macquarie University, Sydney, NSW 2103, Australia; juliuslu2014@gmail.com

* Correspondence: xiangyumeng@jlu.edu.cn

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 intrasensor 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 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.

Keywords: graph neural networks; convolutional neural networks; electroencephalogram classification; intrasensor relationship learning



Citation: Ge, J.; Xu, G.; Lu, J.; Xu, X.; Meng, X. GRAPHSENSOR: A Graph Attention Network for Time-Series Sensor. *Electronics* **2024**, *13*, 2290. <https://doi.org/10.3390/electronics13122290>

Academic Editors: Evangelos Maltezos, Konstantinos Katsaros, Eleftherios Ouzounoglou and Angelos Amditis

Received: 18 April 2024

Revised: 31 May 2024

Accepted: 7 June 2024

Published: 11 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Recent years have seen a greater focus on human behavior recognition, especially monitoring the bhealth 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 [1]. 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 [2], sleep disorders, and brain tumors, while surface electromyography (sEMG) and electrocardiography (ECG) can detect driver drowsiness [3]. In addition, inertial measurement units (IMUs) are used to detect falls, heavy smoking, and alcohol intake [4]. Nowadays, since sensor fusion technology provides a more comprehensive view into people's behavior, graph-based neural networks, such as graph convolutional networks (GCNs) [5] and graph attention networks (GATs) [6], 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 [7], 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 [7,8] (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 [9]. 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. Using the depthwise separable convolution method, a single convolutional filter is applied to each channel, whereas a pointwise convolution creates a linear combination [10].

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) a single segment representation module, and (2) a relationship learning module. Inspired by the vision transformer (ViT) [11], our approach integrates ViT's self-attention mechanisms into the convolutional architecture of the single segment representation module. This integration aims to enhance the global contextual understanding of the model, enabling it to capture both local and long-range dependencies within the sensor data. Such an approach is designed to complement the inherently local focus of traditional CNNs by assessing broader internal relationships and dynamics more effectively.

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 fewer parameters and less training time than the standard approach (more discussions can be found in Section 2.3). In a nutshell, the main contributions of this paper are as follows:

- Our work is one of the first to explore intrasensor 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 fewer parameters and less 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 (<https://github.com/jldx-gjq/Graphnet/tree/main/graphsensor-code-main>).

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. A 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 Section 2.3). We list the notations that are used in this work for convenience (as shown in Table 1).

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 Figure 1) consists of two major parts, signal segment representation $\mathcal{F}_{sr}(\mathcal{S}) = \mathcal{V}$ and relationship learning $\mathcal{G}_{rl}(\mathcal{V}) = \mathcal{V}_{mh}$, where $\mathcal{S}$ is a signal segment set, $\mathcal{V}$ is a feature map, and $\mathcal{V}_{mh}$ represents multi-head features.

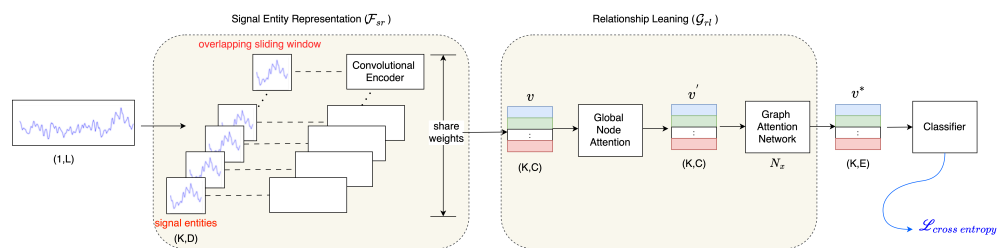


Figure 1. The overall architecture of GRAPHSENSOR. Based on the graph attention network (GAT) framework with specific enhancements.

Given a time series $X = \{x_1, x_2, \dots, x_L\} \in \mathbb{R}^{1 \times L}$, an overlapping sliding window method is used to obtain a signal segment set $\mathcal{S} = \{s_1, s_2, \dots, s_K\} \in \mathbb{R}^{K \times D}$ that is then mapped to $\mathcal{V} = \{v_1, v_2, \dots, v_K\} \in \mathbb{R}^{K \times C}$ by a convolution-based feature encoder $\mathcal{F}_{sr}$, (see the details in Section 2.1). This is followed by the construction of a graph using a graph attention neural network $\mathcal{G}_{rl}$ 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.

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:

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

Signal Segment Representation: Signal segment representation is a convolutional encoder that maps signal segments $\mathcal{S} = \{s_1, s_2, \dots, s_K\} \in \mathbb{R}^{K \times D}$ into a feature space $\mathcal{V} = \{v_1, v_2, \dots, v_K\} \in \mathbb{R}^{K \times C}$. Three 1D convolution layers are used in the convolutional encoder, where a large kernel with the size of 1×49 is used in the first 1D convolution layer for the purpose of boosting CNN's performance by significantly enlarging its effective receptive field [12]. Following that, a smaller-kernel with a size of 1×7 is used on the remaining two layers to further extract the feature map. At the end of the encoder, we utilize a residual squeeze-and-excitation (SE) block [13] 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 as follows:

The convolutional encoder, as shown in Figure 2, starts from a 1D convolution layer with the input $X = \{x_1, x_2, \dots, x_D\}$, $x_i \in \mathbb{R}$, where D is a hyperparameter that represents the length of a signal segment. After that, a max pooling is used to downsample the feature map from $I = \text{Conv1D}_1(X)$, $I \in \mathbb{R}^{A \times D}$ to $I' = \text{Maxpooling}_1(I)$, $I' \in \mathbb{R}^{A' \times D'}$, 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 $\varphi = \text{Maxpooling}_2(\text{Conv1D}_3(\text{Conv1D}_2(I')))$, $\varphi \in \mathbb{R}^{A'' \times D''}$, where A'' is the output size after two 1D convolution layers and max pooling, and D'' is the dimension of the feature output. Afterwards, we utilize a residual squeeze-and-excitation (SE) block [13] 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 φ such that $\varphi' = \text{Conv1D}_5(\text{Conv1D}_4(\varphi))$, $\varphi' \in \mathbb{R}^{B \times M}$, 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 $\phi = \text{Avgpooling}(\varphi')$, $\phi \in \mathbb{R}^{B \times M}$. 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:

$$F = \varphi' \otimes \psi \quad (2)$$

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:

$$O = \sigma(\varphi + F) \quad (3)$$

where σ is a GELU activation function. The output O is the spatial features extracted from the basis segments. Table 2 presents the parameters' value for the convolutional encoder.

Covert signals to spectrogram images: We convert raw signals collected from sensors into spectrogram images, a crucial step in our feature learning process. This conversion includes using the short-time Fourier transform (STFT) to transform time—domain signals into a time—frequency representation. The process involves signal segmentation, application of a window function, Fourier transformation, magnitude spectrum computation, and logarithmic scaling, resulting in spectrograms with the x-axis representing time and the y-axis representing frequency.

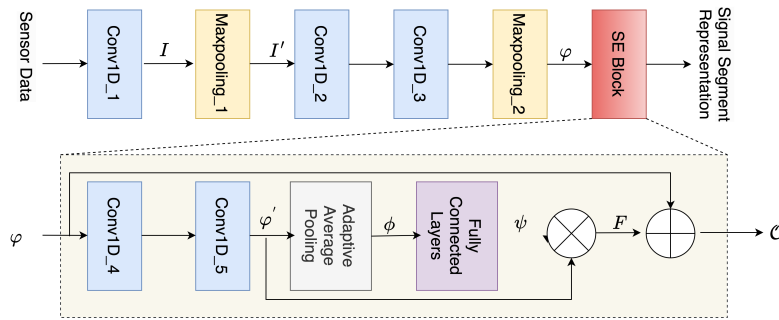


Figure 2. The convolutional encoder in signal segment representation.

Table 2. 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

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 ($\mathbf{R1}$), 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 :

$$W_G = \sigma_1(W_2(\sigma_2(W_1(\text{Pooling}(\mathcal{V})))), \quad (4)$$

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

$$\mathcal{V}' = W_G \times \mathcal{V} \quad (5)$$

As shown in Figure 3, 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 $\mathbb{R}^{\frac{K}{2} \times 1}$. A nonlinear transformation (GELU) is then applied to 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 Equation (5) in Section 2.2.1.

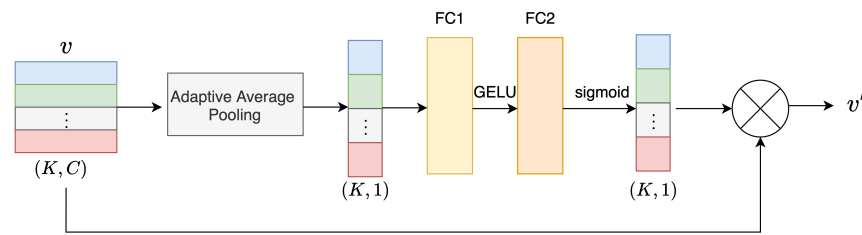


Figure 3. The architecture of the global node attention module.

2.2.2. Graph Attention

The graph attention module has a similar architecture to the global node attention model as shown in Figure 4. 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 1×1 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:

$$v_i^* = \sum_{j \in \mathcal{N}_i} \alpha_{ij} v_j \quad (6)$$

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

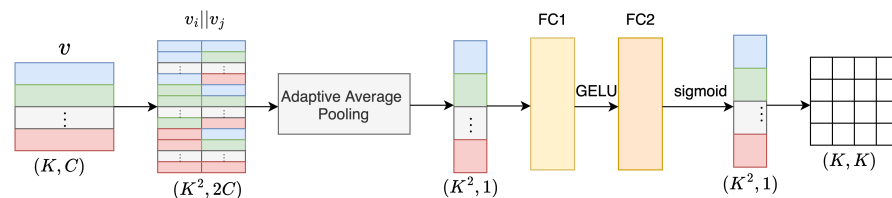


Figure 4. The architecture of graph attention.

1. Graph-based Self-Attention

Adjacency Matrix: A number of previous studies have used Pearson correlations to derive the adjacency matrix [14–16]. 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. Despite the dynamic nature of Pearson correlation, the sensor data often demonstrates periods of relative stability where linear relationships between segments remain consistent, making Pearson correlation a viable method for this analysis. This allows for a straightforward and effective way to measure linear dependencies, essential for our model's ability to map internal relationships accurately. 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. The attention coefficient matrix is usually symmetric; however, most time-series data in the real world are causal in nature [17] and their relationship is therefore antisymmetric. Taking

this characteristic into account, and inspired by [6], our attention mechanism with softmax normalization is expressed as

$$\alpha_{ij} = \frac{\exp(\text{Sigmoid}(\mathbf{a}[\text{pos}_1 \times v_i \parallel \text{pos}_2 \times v_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\text{Sigmoid}(\mathbf{a}[\text{pos}_1 \times v_i \parallel \text{pos}_2 \times v_k]))} \quad (7)$$

where pos_1 and pos_2 are the positional encoding, $\parallel$ is the concatenation operation, and the Sigmoid nonlinearity is applied.

Here, $\mathbf{a}$ 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 Section 2.2.2.

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 [18]. Similarly, we present “1” for the first segment and “2” for the second segment.

2. 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 Figure 5). The self-attention module in this work is split into two blocks: the dense layers and the attention layers [19].

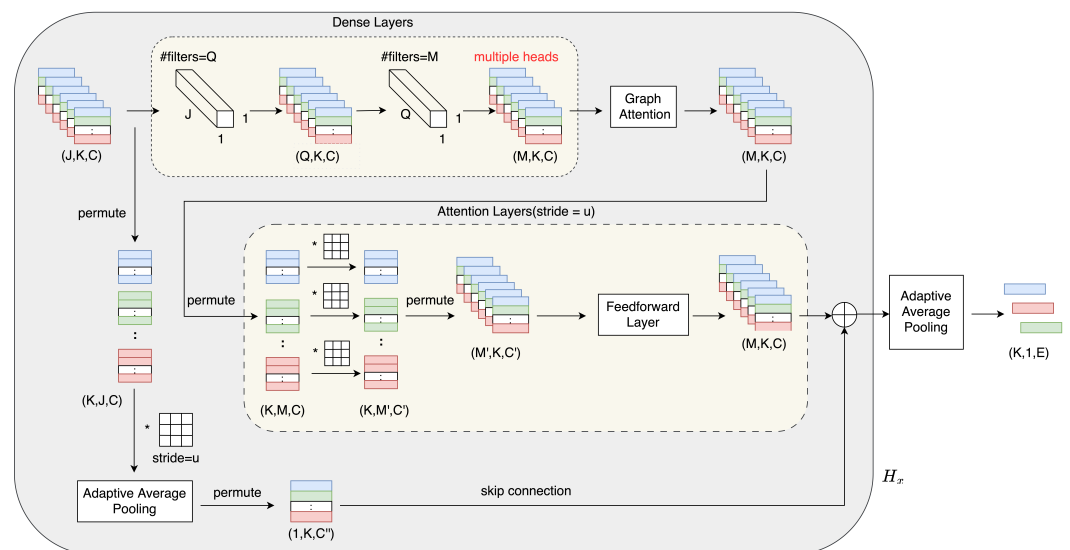


Figure 5. Illustration of the multi-head mechanism.

Dense Layers: Initially, 1×1 convolutions are conducted in the dense layer to project the signal segments to a higher dimension space [6], 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 Figure 5). The expanded dimension is used to generate the independent attention heads.

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 Figure 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 $\#filters > 1$.

Attention Layers: To extract the signal segment features from $\mathcal{V}_{mh}$, 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 [20]).

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. 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.

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 3:

Table 3. 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

2.3. Discussion

Based on prior work [21], 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 Equation (8):

$$MultiHead(standard) = Concat(head_1, ..., head_H)W^O \quad (8)$$

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 Equation (9):

$$MultiHead(ours) = Stack(head_1, ..., head_H) \otimes K \quad (9)$$

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 IN model size (parameters) and training time.

3. Baseline Methods in SLEEP and WISDM

- **DeepSleepNet** [22]: The model exploits a custom CNN architecture followed by an LSTM with a residual connection for sleep stage classification.
- **ResnetLSTM** [23]: The model implements a ResNet architecture for feature extraction, followed by an LSTM to classify EEG signals into different sleep stages.
- **MultitaskCNN** [24]: 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** [25]: This model employs the same CNN architecture as DeepSleepNet [22] followed by an encoder–decoder with attention mechanism.
- **AttnSleepNet** [9]: 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.
- **EfficientNet b4** [26]: EfficientNet b4 utilizes compound scaling to optimize depth, width, and resolution, achieving top-tier accuracy with efficiency in a scalable CNN architecture.
- **ResNet16** [27]: ResNet16 employs a deep 16-layer structure with residual connections to facilitate gradient flow and robust feature learning.
- **MobileNet V3** [28]: MobileNet V3 optimizes for mobile devices using a mix of network architecture search and lightweight convolutions for efficient on-device vision tasks.

4. Experiments

4.1. The Datasets and Implementation Details

Datasets: The experiments are conducted based on two widely used datasets, Sleep-EDF-20 [9,22–25] (SLEEP) (<https://gist.github.com/emadeldeen24/a22691e36759934e53984289a94cb09b>, accessed on 20 July 2022) and WISDM Smartphone and Smartwatch Activity and Biometrics Dataset [29–32] (WISDM) (<https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset+>, accessed on 20 July 2022).

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 Section 4.2.

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.

4.2. Specification for Signal Segment Definition

Table 4 lists the specification of signal segment definition in SLEEP and WISDM datasets.

Table 4. Specification for signal segment definition.

Dataset	L	K	D	P
SLEEP	3000	11	500	0.5
WISDM	200	9	40	0.5

In the SLEEP dataset, each Fpz-Cz channel has a length of 3000 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).

4.3. Comparison with Peer Methods

We evaluate the benefits of our solution against a number of state-of-the-art methods (see the details in Section 3), 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 [9,22–25] used as the baselines. In our methodology, we employ 20-fold cross-validation to ensure a robust estimation of our model’s performance, particularly given the complex nature and the relatively small size of our dataset. This choice is driven by the need to maximally utilize our data for training while allowing thorough validation across different subsets. Each fold serves as a test set exactly once, with the remaining 19 folds used for training, ensuring that each data point is utilized both as part of the training and testing sets. 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 [33–35]) for each signal segment, where x, y, z are the three axes of an accelerometer. 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 [26], ResNet16 [27], and MobileNet V3 [28], 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. The specific parameters are shown as follows in Table 5:

Table 5. Model parameters.

Parameters	Parameter Values
epochs	100
batch-size	64
in-channels	9
num-classes	18
segment-size	40
overlapping-rate	0.5
learning-rate	0.001
dropout	0.5
multiplier	1.0
optimizer	Adam
loss function	CrossEntropyLoss

- **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 [6], and graph attention networks (GAT) [21] 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 heads 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 [21]).

We answer **Q1** using Table 6, which shows the comparison of our method with baseline algorithms on the SLEEP and the WISDM. For the SLEEP dataset, our GraphSensor model achieves an impressive accuracy of 89.4%, significantly higher than many of the grid representations used by CNN and LSTM-based algorithms [22–25], and the SOTA performance provided by AttnSleepNet [9]. Notably, GRAPHSENSOR maintains this high accuracy while also demonstrating a fast inference time of only 0.56 s. This is particularly remarkable given that the methods with closer accuracy, such as AttnSleepNet, have a longer inference time (0.63 s), and those with shorter inference times, like SleepEEGNet (0.47 s), have considerably lower accuracy (81.5%). The efficiency of GRAPHSENSOR can be attributed to the proposed convolution-based multi-head attention mechanism, which allows the heads to be expanded with fewer parameters, as discussed in Section 4.4 of the manuscript. This design choice not only aids in reducing the model size (parameters) but also cuts down on training time, enabling the model to process information rapidly without compromising accuracy.

Table 6. The performance comparison of the state-of-the-art approaches.

Datasets	Methods	Accuracy	Inference Time
SLEEP	DeepSleepNet [22]	81.9	1.28
	ResnetLSTM [23]	82.5	1.09
	MultitaskCNN [24]	83.1	0.58
	SleepEEGNet [25]	81.5	0.47
	AttnSleepNet [9]	84.4	0.63
	<i>Ours</i>	89.4	0.56
WISDM	EfficientNet b4 [26]	27.0	0.26
	ResNet16 [27]	55.5	0.19
	MobileNet V3 [28]	48.8	0.12
	<i>Ours</i>	69.3	0.10

On the WISDM dataset, GRAPHSENSOR establishes its dominance with an accuracy of 69.3% and an inference time of only 0.10 s, greatly surpassing other models such as EfficientNet b4 (27.0% accuracy, 0.26 s inference time), ResNet16 (55.5% accuracy, 0.19 s inference time), and MobileNet V3 (48.8% accuracy, 0.12 s inference time). 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 substantial lead in accuracy and the minimal inference time underscore GRAPHSENSOR's capability to capture the intricate patterns within the IMU signals effectively. Thank you once again for your suggestions.

The response to **Q2** is given in Table 7, 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 [6] 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 lower 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 [36] on GRAPHSENSOR and conducting real-world experiments are our future work.

Table 7. The performance comparison of the multi-head approaches (OM: GPU out of memory, #par: Model’s number of parameters, Error: run-time error).

Methods	32 Heads (#par)	128 Heads (#par)	512 Heads (#par)
Transformer [21]	86.3 (2098K)	87.1 (3311K)	Error
GAT [6]	85.4 (31,897K)	86.4 (125,416K)	OM (499,494K)
Ours	87.3 (1177K)	89.4 (1713K)	90.1 (7033K)

4.4. Ablation Study

To further investigate the effectiveness of our solution, we conduct an ablation study on the SLEEP dataset, as shown in Table 8. 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?**
- **Q7: How does GRAPHSENSOR excel in relationship learning?**
- **Q8: How does GRAPHSENSOR generalize on other datasets?**

For Q3, from Table 8, 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.

Table 8. Ablation and Sensitivity Study.

Questions	Methods	Accuracy	#Parameters	Time Cost
Q3	GRAPHSENSOR (1000, 5)	87.4	1253K	≈1.0 h
	GRAPHSENSOR (800, 6)	87.5	1323K	≈1.0 h
	GRAPHSENSOR (600, 9)	88.9	1545K	≈1.7 h
	GRAPHSENSOR (500, 11)	89.4	1713K	≈1.7 h
Q4	GRAPHSENSOR (500, 11) with head number 16	85.9	826K	≈0.6 h
	GRAPHSENSOR (500, 11) with head number 32	87.3	1177K	≈0.6 h
	GRAPHSENSOR (500, 11) with head number 128	89.4	1713K	≈1.7 h
	GRAPHSENSOR (500, 11) with head number 512	90.1	7033K	≈3.0 h

The results of **Q4** in Table 8 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 the 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% (five random permutation rounds), demonstrating that GRAPHSENSO provides valuable information regarding temporal aspects.

For **Q6**, we perform a stress test on the WISDM dataset and SLEEP dataset (as shown in Tables 9 and 10), rearranging the length of the samples from $n = 200$ (10 s) to $n = 20$ (1 s) and $n = 500$ (10 s) to $n = 50$ (1 s).

Table 9. Stress test on the WISDM dataset.

Datasets	Methods	Accuracy	Sample Length (n)
WISDM	EfficientNet b4 [26]	27.0	200
	ResNet16 [27]	55.5	200
	MobileNet V3 [28]	48.8	200
	Ours	69.3	200
	Ours	56.1	20

Table 10. Stress test on the SLEEP dataset.

Datasets	Methods	Accuracy	Sample Length (n)
SLEEP	DeepSleepNet [22]	81.9	500
	ResnetLSTM [23]	82.5	500
	MultitaskCNN [24]	83.1	500
	SleepEEGNet [25]	81.5	500
	AttnSleepNet [9]	84.4	500
	Ours	89.4	500
	Ours	81.4	50

For the WISDM dataset, our method 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%. The results on the SLEEP dataset are even more compelling. With a sample length of 500, GraphSensor achieves an accuracy of 89.4%, which is notably higher than that of other methods. Under stress conditions with a reduced sample length of 50, GraphSensor maintains an accuracy of 81.4%. This is a clear indication of its robustness, as the performance remains close to that

of the other models tested under optimal, non-stressed conditions. 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.

To address Q7, we visualize the internal relationship using Pearson correlation, as shown in Figures 6 and 7.

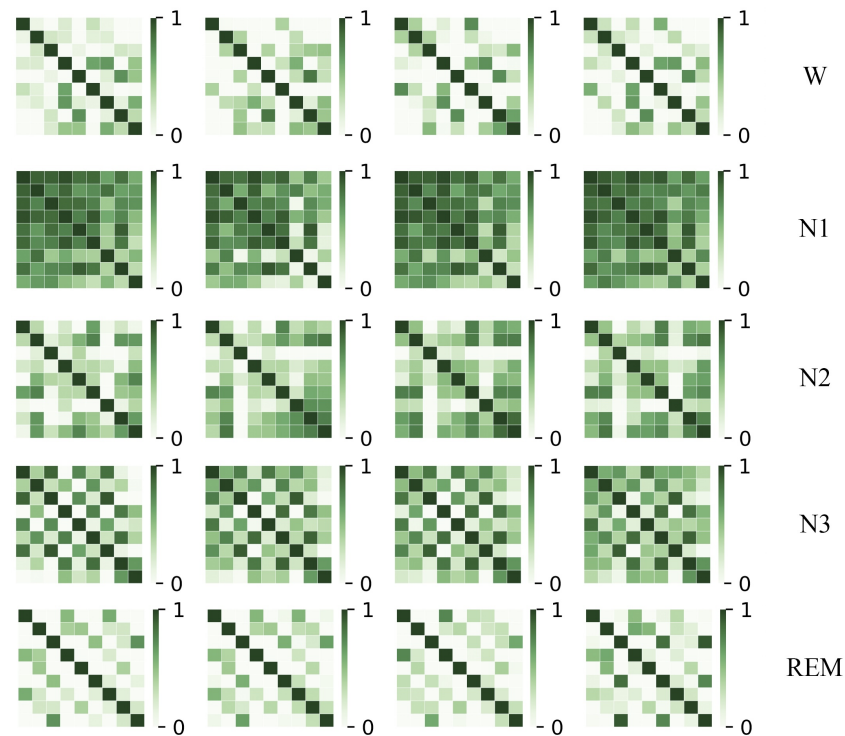


Figure 6. Illustration of the internal relationship using Pearson correlation (SLEEP).

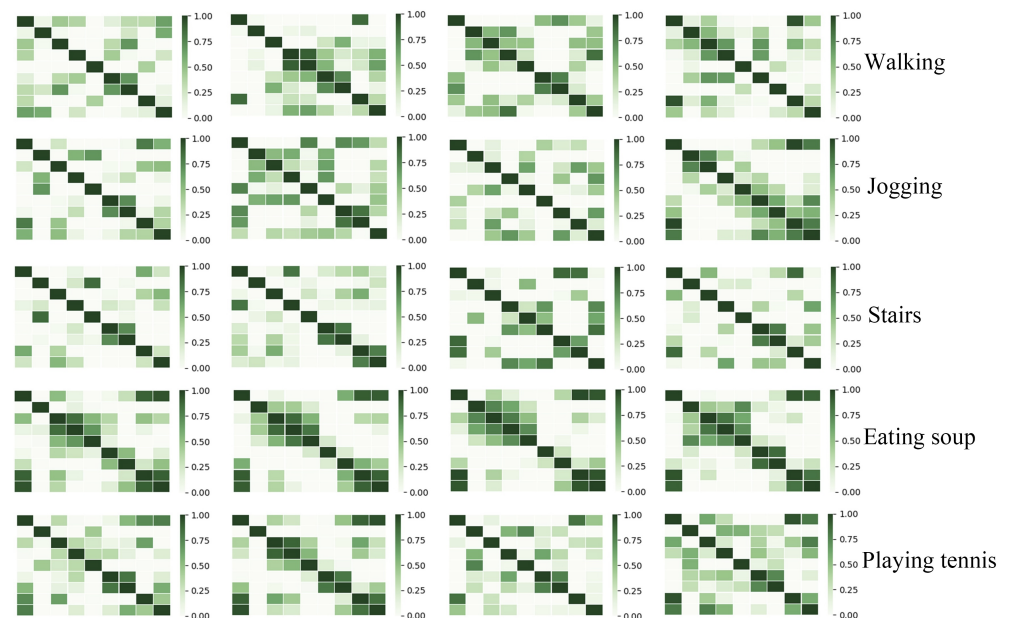


Figure 7. The internal relationship using Pearson correlation (WISDM).

SLEEP: There are five different stages of sleep (W, N1, N2, N3, REM) in SLEEP. The heatmaps shown in Figure 6 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 a different batch representing different

signal segments for the same sleep stage. As shown in Figure 6, in each label, all the signal segments exhibit similar heatmap patterns but with distinct difference with signal segments from different labels (even from the 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 shown in Figure 7 depict the internal relationship between five activities (walking, jogging, climbing stairs, eating soup, and playing tennis) using Pearson correlation. From Figure 7, we can observe that eating soup and playing tennis have unique patterns, while walking, jogging, and climbing stairs, as three confounding activities, have similar intrasensor relationships.

For Q8, to further validate the generalization of GRAPHSENSOR, we have incorporated a new dataset, BCIC-IV-2a [37], into our experimental evaluation. BCIC-IV-2a features nine subjects in a four-class MI-EEG task, recorded via 22 electrodes and three EOG channels at 250 Hz. The training and evaluation sessions comprise six runs of 12 trials for each class, totaling 288 trials per subject. The BCIC-IV-2a dataset comprises 55,000 (22×2500) samples, each composed of data from 22 channels. Within each channel, there are 2500 data points, amounting to 10 s of recorded data per channel. In our experimental evaluation, GRAPHSENSOR is benchmarked against several baseline techniques. These baseline methods for MI-EEG classification can be categorized into two groups: traditional frequency-domain feature extraction approaches and models based on Riemannian manifolds. The traditional methods, such as FBCSP [38], ConvNet [39], and EEGNet [40], extract features by leveraging the temporal and spatial frequency characteristics of EEG signals, with performance comparable to each other. FBCNet [41], an extension of these methods, achieves superior performance by integrating both frequency and spatial patterns, utilizing bandpass filters adept at capturing frequency information. On the other hand, models based on Riemannian manifolds, including MDM [42], TSM [43], and SPDNet [44], utilize geometric quantities and distances on SPD manifolds for EEG classification. However, their performance, particularly on the MI-KU dataset, is relatively poor compared to traditional methods, indicating limitations in employing geometric quantities for high-level classification features. Notably, Graph-CSPNet [45] stands out with its refined time–frequency segmentation technique, offering more accurate characterization of EEG signals and demonstrating superior performance compared to both traditional and Riemannian-based approaches. The results are shown as Table 11:

Table 11. The performance comparison on BCIC-IV-2a dataset.

Datasets	Methods	Accuracy
BCIC-IV-2a	FBCSP [38]	66.13
	ConvNet [39]	81.9
	EEGNet [40]	82.5
	FBCNet [41]	83.1
	MDM [42]	50.74
	TSM [43]	49.72
	SPDNet [44]	55.67
	Graph-CSPNet [45]	81.5
	Ours	84.3

The results show that GRAPHSENSOR achieves the highest accuracy at 72.48%. This indicates that GRAPHSENSOR is proficient at handling the complexity of motor imagery EEG data, surpassing the capabilities of other advanced deep learning models such as FBCSP, ConvNet, EEGNet, and even the closely related FBCNet and Graph-CSPNet. This comparative analysis underscores the capability of GRAPHSENSOR to adeptly handle complex EEG data, indicating its robustness and generalization ability across various subjects and conditions.

5. Related Work

Feature Representation: The feature representation capabilities of convolutional neural networks (CNNs) are well-documented across various applications, particularly in object classification tasks. CNNs operate by sliding filters over input data, performing element-wise matrix multiplications at each position, and summing the results to produce a feature map. This convolutional operation captures local dependencies within the input data, making CNNs highly effective for images and similar structured data. Recent advancements in CNN architectures have further enhanced their effectiveness. ResNet, introduced by He et al. [46], employs residual connections to facilitate training deeper networks by allowing gradients to flow through an alternative shortcut path. EfficientNet, developed by Tan and Le [26], scales up CNNs in a more structured manner to achieve better accuracy and efficiency.

In pursuit of more efficient computation, Chollet [10] proposed a modification known as depthwise separable convolution, which is utilized extensively in MobileNet models [20]. This approach divides the convolution process into two layers: depthwise convolution that applies a single filter per input channel, and pointwise convolution that combines the outputs of the depthwise layer across channels. This method reduces computational cost and model size, making it suitable for mobile and edge devices. Despite these advantages, depthwise separable convolutions generally assume independence between the channels during the depthwise phase, which may not effectively capture the complex interchannel dependencies necessary for some applications, especially those involving intricate signal processing tasks.

Parallel to CNNs, recurrent neural networks (RNNs) offer a different approach to feature representation, especially for time-series data where temporal dependencies are crucial. Models such as LSTM [47] and BiLSTM [48] are particularly adept at learning important features over time and forgetting non-essential information, thanks to their gated memory mechanisms. Unlike CNNs, RNNs process data sequentially, making them ideal for problems where the order of data points is important, such as speech and language processing. The directional nature of standard RNNs, however, limits their ability to learn complex relational patterns in data, a limitation addressed by more sophisticated architectures like graph neural networks (GNNs) [49], which consider the data as a graph to capture pairwise relationships among elements.

While CNNs are preferred for spatial data due to their hierarchical feature extraction, RNNs excel in domains requiring memory of past inputs. Depthwise separable convolutions offer a middle ground by providing efficiency and moderate relational learning between input channels. Our work on Graphnet builds upon these foundational concepts by integrating graph-based learning to capture both spatial and temporal relationships, addressing the limitations of each approach and providing a comprehensive framework for feature representation.

Relationship Learning: The intricate nature of relationships in time-series data, especially their non-Euclidean structure, necessitates a representation that can effectively capture the complex interdependencies inherent in such data. Graph neural networks (GNNs) with attention mechanisms are particularly well-suited for this purpose due to their ability to model data as a graph, allowing for a dynamic representation of relationships.

Graph-Based Neural Networks: Graph attention networks (GATs), introduced by [6], are a prime example of the effective use of attention mechanisms within GNNs. GATs compute attention scores via feedforward neural networks, which adaptively weigh the significance of each node's contributions to its neighbors, facilitating selective focus on more relevant features. This capability is crucial for analyzing time-series data where the relevance of specific data points can vary significantly. Comparatively, other GNN variants, such as Graph convolutional networks (GCN) [50] and GraphSAGE [51], compute attention scores based on the cosine similarity (dot product), focusing on aggregating neighborhood features through a more static approach. While effective, these methods

lack the adaptability of GATs in focusing dynamically on parts of the graph that are more relevant to the specific learning task.

Asymmetry in Time-Series Data: The majority of real-world time-series data exhibit causal and asymmetric relationships, where the direction of influence between data points matters significantly. GAT's ability to model these asymmetric relationships makes it particularly suitable for time-series analysis. This is because its attention mechanism inherently accounts for the directionality in its computations, aligning closely with the causal nature of the data.

Utilizing Graph Attention in Single-Sensor Data: In our research, we extend the application of graph-based learning to single-sensor data analysis. Unlike multi-sensor approaches such as GraphSleepNet [52], which learns external relationships across multiple sensors, our focus is on uncovering intricate internal patterns within data from a single sensor. To this end, we also explore the use of the original multi-head attention mechanism from the transformer architecture by Vaswani et al. [21]. This mechanism, which combines multiple attention heads into a single operational framework, allows for a richer and more nuanced feature representation by capturing various aspects of the data in parallel.

Our baseline comparisons include both GAT and transformer architectures to benchmark our model's performance against state-of-the-art methods in graph-based learning. While both approaches are built on the foundation of attention mechanisms, their implementation and focus differ significantly, providing a comprehensive view of the potential of graph-based learning in handling complex time-series data.

6. Conclusions

In this paper, we presented GRAPHSENSOR, a novel deep graph attention network tailored for analyzing the internal relationships among signal segments from a single sensor. This model innovatively applies a convolution-based multi-head graph attention network to efficiently construct and learn the graph structure representing pairwise connections between signal segments. Our validation across three distinct datasets shows that GRAPHSENSOR not only achieves superior performance compared to existing benchmarks but also significantly improves efficiency, achieving at least a 50% reduction in model size over traditional multi-head solutions while maintaining higher accuracy. These advancements underscore GRAPHSENSOR's potential in precisely modeling time-series data for sensor-based applications, setting new standards in graph-based signal analysis and pushing the boundaries of what is possible in understanding complex sensor-generated data.

Author Contributions: Conceptualization, J.G. and G.X.; methodology, J.L.; validation, J.G. and X.X.; writing—original draft preparation, J.G. and J.L.; writing—review and editing, G.X. and X.M.; visualization, J.L. and X.X.; supervision, G.X.; funding acquisition, X.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Jilin Province Science and Technology Development Plan Project under Grants 20200401076GX, and the Jilin Provincial Department of Science and Technology Innovation Platform (Base) and Talent Special Project 20220508043RC.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sharma, M.; Majumdar, P. Occupational lifestyle diseases: An emerging issue. *Indian J. Occup. Environ. Med.* **2009**, *13*, 109. [\[CrossRef\]](#)
2. Siddiqui, M.M.; Rahman, S.; Saeed, S.H.; Banodia, A. EEG signals play major role to diagnose sleep disorder. *Int. J. Electron. Comput. Sci. Eng. (IJECSSE)* **2013**, *2*, 503–505.
3. Lu, J.; Zheng, X.; Tang, L.; Zhang, T.; Sheng, Q.Z.; Wang, C.; Jin, J.; Yu, S.; Zhou, W. Can Steering Wheel Detect Your Driving Fatigue? *IEEE Trans. Veh. Technol.* **2021**, *70*, 5537–5550. [\[CrossRef\]](#)
4. Rokni, S.A.; Nourollahi, M.; Ghasemzadeh, H. Personalized human activity recognition using convolutional neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LO, USA, 2–7 February 2018; Volume 32.

5. Duvenaud, D.K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R.P. Convolutional networks on graphs for learning molecular fingerprints. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2224–2232.
6. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 10 April–3 May 2018.
7. Morales, S.; Bowers, M.E. Time-frequency analysis methods and their application in developmental EEG data. *Dev. Cogn. Neurosci.* **2022**, *54*, 101067. [\[CrossRef\]](#)
8. Liu, X.; Hong, D.; Chanussot, J.; Zhao, B.; Ghamisi, P. Modality translation in remote sensing time series. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 5401614. [\[CrossRef\]](#)
9. Eldele, E.; Chen, Z.; Liu, C.; Wu, M.; Kwok, C.K.; Li, X.; Guan, C. An Attention-Based Deep Learning Approach for Sleep Stage Classification with Single-Channel EEG. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2021**, *29*, 809–818. [\[CrossRef\]](#)
10. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
11. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
12. Ding, X.; Zhang, X.; Zhou, Y.; Han, J.; Ding, G.; Sun, J. Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs. *arXiv* **2022**, arXiv:2203.06717.
13. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
14. Sun, Y.; Yin, Q.; Fang, R.; Yan, X.; Wang, Y.; Bezerianos, A.; Tang, H.; Miao, F.; Sun, J. Disrupted functional brain connectivity and its association to structural connectivity in amnesic mild cognitive impairment and Alzheimer’s disease. *PLoS ONE* **2014**, *9*, e96505. [\[CrossRef\]](#)
15. Tijms, B.M.; Yeung, H.M.; Sikkes, S.A.; Möller, C.; Smits, L.L.; Stam, C.J.; Scheltens, P.; van der Flier, W.M.; Barkhof, F. Single-subject gray matter graph properties and their relationship with cognitive impairment in early- and late-onset Alzheimer’s disease. *Brain Connect.* **2014**, *4*, 337–346. [\[CrossRef\]](#)
16. Xue, H.; Peng, J.; Shang, X. Towards gene function prediction via multi-networks representation learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 10069–10070.
17. Dahlhaus, R.; Eichler, M. Causality and graphical models in time series analysis. In *Oxford Statistical Science Series*; Oxford Science Publication: Oxford, UK, 2003; pp. 115–137.
18. Wang, B.; Zhao, D.; Lioma, C.; Li, Q.; Zhang, P.; Simonsen, J.G. Encoding word order in complex embeddings. *arXiv* **2019**, arXiv:1912.12333.
19. Pan, X.; Ge, C.; Lu, R.; Song, S.; Chen, G.; Huang, Z.; Huang, G. On the integration of self-attention and convolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 815–825.
20. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
21. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
22. Supratak, A.; Dong, H.; Wu, C.; Guo, Y. DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2017**, *25*, 1998–2008. [\[CrossRef\]](#)
23. Sun, Y.; Wang, B.; Jin, J.; Wang, X. 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, Beijing, China, 13–15 October 2018; pp. 1–5. [\[CrossRef\]](#)
24. Phan, H.; Andreotti, F.; Cooray, N.; Chen, O.Y.; De Vos, M. Joint classification and prediction CNN framework for automatic sleep stage classification. *IEEE Trans. Biomed. Eng.* **2018**, *66*, 1285–1296. [\[CrossRef\]](#)
25. Mousavi, S.; Afghah, F.; Acharya, U.R. SleepEEGNet: Automated sleep stage scoring with sequence to sequence deep learning approach. *PLoS ONE* **2019**, *14*, e0216456. [\[CrossRef\]](#)
26. Tan, M.; Le, Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
27. Ismail Fawaz, H.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.A.; Petitjean, F. Inceptiontime: Finding alexnet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [\[CrossRef\]](#)
28. Gupta, S. Deep learning based human activity recognition (HAR) using wearable sensor data. *Int. J. Inf. Manag. Data Insights* **2021**, *1*, 100046. [\[CrossRef\]](#)
29. Weiss, G.M.; Yoneda, K.; Hayajneh, T. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access* **2019**, *7*, 133190–133202. [\[CrossRef\]](#)
30. Mekruksavanich, S.; Jitpattanakul, A.; Youplao, P.; Yupapin, P. Enhanced hand-oriented activity recognition based on smartwatch sensor data using lstms. *Symmetry* **2020**, *12*, 1570. [\[CrossRef\]](#)

31. Mekruksavanich, S.; Jitpattanakul, A. Smartwatch-based human activity recognition using hybrid lstm network. In Proceedings of the 2020 IEEE SENSORS, Rotterdam, The Netherlands, 25–28 October 2020; pp. 1–4.
32. Singh, S.P.; Sharma, M.K.; Lay-Ekuakille, A.; Gangwar, D.; Gupta, S. Deep ConvLSTM with self-attention for human activity decoding using wearable sensors. *IEEE Sens. J.* **2020**, *21*, 8575–8582. [\[CrossRef\]](#)
33. Lu, J.; Zheng, X.; Sheng, M.; Jin, J.; Yu, S. Efficient human activity recognition using a single wearable sensor. *IEEE Internet Things J.* **2020**, *7*, 11137–11146. [\[CrossRef\]](#)
34. Bhandari, B.; Lu, J.; Zheng, X.; Rajasegarar, S.; Karmakar, C. Non-invasive sensor based automated smoking activity detection. In Proceedings of the 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jeju Island, Republic of Korea, 11–15 July 2017; pp. 845–848.
35. Chow, D.H.K.; Tremblay, L.; Lam, C.Y.; Yeung, A.W.Y.; Cheng, W.H.W.; Tse, P.T.W. Comparison between Accelerometer and Gyroscope in Predicting Level-Ground Running Kinematics by Treadmill Running Kinematics Using a Single Wearable Sensor. *Sensors* **2021**, *21*, 4633. [\[CrossRef\]](#)
36. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv* **2017**, arXiv:1710.09282.
37. Tangermann, M.; Müller, K.R.; Aertsen, A.; Birbaumer, N.; Braun, C.; Brunner, C.; Leeb, R.; Mehring, C.; Mueller-Putz, G.; Nolte, G.; et al. Review of the BCI competition IV. *Front. Neurosci.* **2012**, *6*, 21084. [\[CrossRef\]](#)
38. Ang, K.K.; Chin, Z.Y.; Zhang, H.; Guan, C. Filter bank common spatial pattern (FBCSP) in brain-computer interface. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 2390–2397.
39. Liu, Y.; Shao, H.; Bai, B. A Novel Convolutional Neural Network Architecture with a Continuous Symmetry. In Proceedings of the CAAI International Conference on Artificial Intelligence, Fuzhou, China, 22–23 July 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 310–321.
40. Lawhern, V.J.; Solon, A.J.; Waytowich, N.R.; Gordon, S.M.; Hung, C.P.; Lance, B.J. EEGNet: A compact convolutional neural network for EEG-based brain–computer interfaces. *J. Neural Eng.* **2018**, *15*, 056013. [\[CrossRef\]](#)
41. Mane, R.; Chew, E.; Chua, K.; Ang, K.K.; Robinson, N.; Vinod, A.P.; Lee, S.W.; Guan, C. FBCNet: A multi-view convolutional neural network for brain-computer interface. *arXiv* **2021**, arXiv:2104.01233.
42. Tevet, G.; Raab, S.; Gordon, B.; Shafir, Y.; Cohen-Or, D.; Bermano, A.H. Human motion diffusion model. *arXiv* **2022**, arXiv:2209.14916.
43. Lin, J.; Gan, C.; Han, S. Temporal shift module for efficient video understanding. In Proceedings of the 2019 IEEE International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019. [\[CrossRef\]](#)
44. Huang, Z.; Van Gool, L. A riemannian network for spd matrix learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
45. Ju, C.; Guan, C. Graph Neural Networks on SPD Manifolds for Motor Imagery Classification: A Perspective From the Time–Frequency Analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**. [\[CrossRef\]](#)
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
47. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Networks Learn. Syst.* **2016**, *28*, 2222–2232. [\[CrossRef\]](#)
48. Graves, A.; Fernández, S.; Schmidhuber, J. Bidirectional LSTM networks for improved phoneme classification and recognition. In Proceedings of the International Conference on Artificial Neural Networks, Warsaw, Poland, 11–15 September 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 799–804.
49. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [\[CrossRef\]](#)
50. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
51. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1025–1035.
52. Jia, Z.; Lin, Y.; Wang, J.; Zhou, R.; Ning, X.; He, Y.; Zhao, Y. GraphSleepNet: Adaptive Spatial-Temporal Graph Convolutional Networks for Sleep Stage Classification. In Proceedings of the International Joint Conference on Artificial Intelligence, Yokohama, Japan, 11–17 July 2020; pp. 1324–1330.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.