Edge-Aware Graph Attention Network for Ratio of Edge-User Estimation in Mobile Networks

Jiehui Deng[†], Sheng Wan[†], Xiang Wang[‡], Enmei Tu[‡],
Xiaolin Huang[‡], Jie Yang[‡] and Chen Gong[†§*]
[†] PCA Lab, the Key Laboratory of Intelligent Perception and Systems for High-Dimensional
Information of Ministry of Education, School of Computer Science and Engineering,
Nanjing University of Science and Technology, Nanjing, China
[‡]Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China
[§]Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR, China
Email: chen.gong@njust.edu.cn

***Abstract*—Estimating the Ratio of Edge-Users (REU) is an important issue in mobile networks, as it helps the subsequent adjustment of loads in different cells. However, existing approaches usually determine the REU manually, which are experience-dependent and labor-intensive, and thus the estimated REU might be imprecise. Considering the inherited graph structure of mobile networks, in this paper, we utilize a graph-based deep learning method for automatic REU estimation, where the practical cells are deemed as nodes and the load switchings among them constitute edges. Concretely, Graph Attention Network (GAT) is employed as the backbone of our method due to its impressive generalizability in dealing with networked data. Nevertheless, conventional GAT cannot make full use of the information in mobile networks, since it only incorporates node features to infer the pairwise importance and conduct graph convolutions, while the edge features that are actually critical in our problem are disregarded. To accommodate this issue, we propose an Edge-Aware Graph Attention Network (EAGAT), which is able to fuse the node features and edge features for REU estimation. Extensive experimental results on two real-world mobile network datasets demonstrate the superiority of our EAGAT approach to several state-of-the-art methods.**

I. Introduction

In mobile networks, estimating the Ratio of Edge-Users (REU) [1] according to the properties of different cells (*e.g.*, residential areas and working areas, etc) and edges is a very important issue as it offers a crucial cue for balancing the load of different cells. Here "REU" means the proportion of users with less than 5 Million bits per second (Mbps) network speed in all users within a cell, which is actually a measurement of the ratio of low-throughput users. Usually, The users sharing the network resources in a cell may have different throughput. The users with high-throughput are generally satisfied with the network speed. While for low-throughput users, a poor network speed may bring troubles to their regular work and life. For example, in the working daytime from Monday to Friday, the REU in working area will be high, and thus more bandwidth or other network resources should be provided to the working areas to satisfy the requirements of the inside users. On the contrary, for the periods in holidays such as Saturday and Sunday, the REU in residential area will be high as most people will stay at home, so the connection speed for these users is expected to be accelerated.

[*]Corresponding author.

In current practical implementations, engineers simply determine the REU manually based on their previous experiences, so it is experience-dependent and labor-intensive, and thus the estimated REU might be inaccurate [2], [3]. Therefore, in this paper, we aim to use a machine learning technique to adaptively estimate or predict the REU in a future time according to the past numerical records of network structures and REUs, so that the network load can be better adjusted in advance. Note that the mobile network naturally contains a graph structure where different cells are nodes and the load switching among them constitutes edges, therefore by regarding historical REU as the target value of each node, the training of an REU estimation model can be directly formulated as a node regression problem. Currently, there mainly exist two classical solutions to cope with node regression, namely, matrix factorization [4], [5] and random walk [6], [7]. However, since our task requires a generalizable inductive model [8] that can predict the REU of nodes in an unseen graph, so these two methods are not applicable here. This is because the above two methods belong to transductive learning without the ability to do node inference in previously unseen cases. Consequently, in this paper, we propose to employ the Graph Convolutional Network (GCN) with good generalizability to serve as the backbone of our regression model.

Recently, GCN-based methods [9] have received intensive attention and achieved satisfactory results in several applications, such as image classification [10], [11], [12], [13], text classification [14], [15], [16], biological networks [17], [18], [19], transportation networks [20], [21], [22], and citation networks [9], [23], [24]. It is notable that GCN learns a function for aggregating information from neighborhoods of every node in a graph, and thus representative embeddings of nodes can be acquired which further leads to satisfactory learning results. Moreover, some of the GCN models, *e.g.* [23], can be naturally applied to unseen graphs and are demonstrated to be effective in both inductive learning and transductive learning. Lastly, intensive experiments have demonstrated that the GCN-based models often outperform the matrix decomposition and random walk-based methods on various tasks [23], [25].

Nevertheless, as mentioned above, the graph edges and their features also play a critical role in determining the REU in mobile networks. For example, Cell Individual Offset (CIO) [26] evaluates the load switching tendency between two cells and is practically verified to be strongly related to the output REU. Unfortunately, this information regarding edge features cannot be directly utilized by conventional GCN. Therefore, based on Graph Attention Network (GAT) [15], this paper proposes an Edge-Aware Graph Attention Network (EAGAT) which fuses the node features and edge features in a principled way. Specifically, the proposed EAGAT employs attention-based architecture to perform node regression. In the training stage, as the value of the CIO is related to the connection between cells, our methodology takes both the node features and the edge features into consideration to compute the attention coefficients between graph nodes, where self-attention strategy is employed at the same time. Since the edge information is additionally incorporated, the learned attention coefficients can be more accurate than those of conventional GAT. After that, feature aggregation is performed according to the refined attention coefficients. Finally, the deviation between the REU output by our network and the expected value is reduced by minimizing their Mean Squared Error (MSE) [27]. In the test stage, we directly apply the well-trained EAGAT model to a new graph and obtain the predicted REU on the nodes of this graph.

In fact, currently, there are some preliminary researches on learning with a graph with edge features. For example, Decagon [18] attempts to learn different sets of parameters for various edge types (*e.g.*, Gastrointestinal bleed and Bradycardia side effect), and uses the learned embeddings to predict the side effect of drug combinations. Edge2vec [19] aims to identify biological entities such as genes, proteins, drugs, diseases problem and learns a transition probability matrix among the edge types with EM algorithm [28]. However, these two methods can only be utilized to process graphs with the edge types representing different relationships between nodes. The work [29] proposes a method named as EGNN, which takes both the node features and edge features as inputs to produce pairwise attention coefficients for neighbor aggregation. However, the edge features here are merely utilized as a binary indication of whether there is a connection between the two nodes. Therefore, existing works cannot adequately deploy the edge features in our case and thus is not suitable for the investigated REU estimation problem.

In summary, this paper has made the following technical contributions:

- In view of the inherited graph structure of mobile networks, we employ a GCN-based framework to fit the mobile network data and estimate REU parameters.
- Inspired by GAT, the node features and the edge features are encouraged to work collaboratively for learning the refined pairwise importance among graph nodes.
- Extensive experimental results on two real-world mobile network datasets demonstrate the effectiveness of our EAGAT method.

II. Related Work

In recent years, there has been increasing interest in extending convolution to graph structures. Here, we review some representative works on GCN in this section. As introduced in [30] and [31], the advanced works in this area can be categorized as spectral-based methods and spatial-based methods, respectively.

A. Spectral-Based Graph Convolution

The first notable spectral-based GCN was developed by Bruna *et al.* [32], where the convolution operation is defined in the Fourier domain by computing the eigen-decomposition of the graph Laplacian [33], [34], [35]. This operation can also be expressed as a signal x filtered by $g_\theta = diag(\theta)$ parameterized by θ in the Fourier domain, namely

$$g_\theta \star x = \mathbf{U} g_\theta \mathbf{U}^\top x, \tag{1}$$

where $\mathbf{U}$ is the matrix composed of the eigenvectors of the normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. Here $\mathbf{D}$ is the degree matrix, $\mathbf{I}$ denotes the identity matrix, $\mathbf{A}$ denotes the adjacency matrix of the graph, and $\mathbf{\Lambda}$ is a diagonal matrix of which the diagonal elements are the eigenvalues of $\mathbf{L}$. Henaff *et al.* [36] employ a parameterization with smooth coefficients which aims to make the spectral filters spatially localized. Then, Defferrard *et al.* [37] propose to utilize Chebyshev polynomials and its approximate evaluation scheme to reduce the computational cost, which leads to the localized filtering. Recently, Kipf *et al.* [9] simplify the previous methods by showing the first-order approximation to the Chebyshev polynomials as the graph filter spectrum, which greatly restrains the number of parameters and alleviates the over-fitting problem. Although the spectral-based methods are capable of conducting convolution operation on graphs, the learned filters depend on the graph structure. As a result, the model trained under a specific structure usually cannot be directly applied to other unseen graphs [31].

B. Spatial-Based Graph Convolution

Considering the poor generalization ability of spectral-based models to new graphs, spatial-based models perform graph convolution directly on spatially close nodes. For instance, Duvenaud *et al.* [38] utilize different weight matrices for nodes with different degrees. Nevertheless, it cannot be applied to large-scale graphs with large node degrees, because many weight matrices need to be learned at each layer, which may lead to the overfitting problem. Atwood *et al.* [14] propose a diffusion-based graph convolutional network which uses transition matrices to define the neighborhoods for nodes when learning weights for each input channel and neighborhood degree. Niepert *et al.* [39] extract and normalize neighborhoods which contains a fixed number of nodes to address the issue.

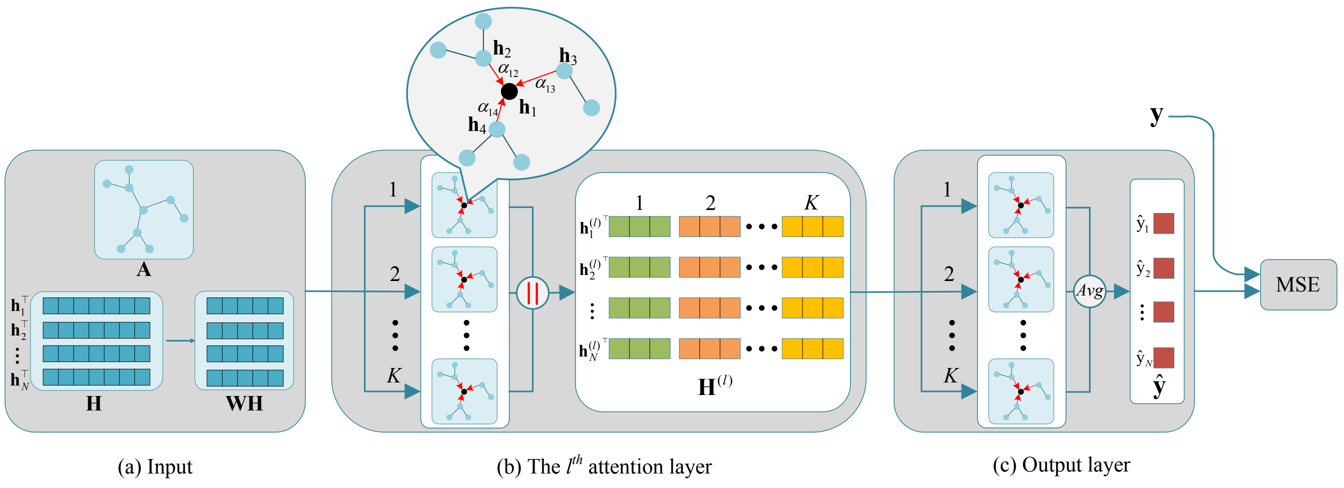

Fig. 1. The training pipeline of our algorithm. (a) presents the input of the network, including the adjacency matrix $\mathbf{A}$ and node features $\mathbf{H} = [\mathbf{h}_1^\top; \mathbf{h}_2^\top; \ldots; \mathbf{h}_N^\top]$, where N is the number of nodes. After that, the weight matrix of a shared linear transformation $\mathbf{W}$ is applied to each node; (b) indicates the l^{th} multi-head attention layer which adopts the hidden representations $\mathbf{H}^{(l)} = \left[\mathbf{h}_1^{(l)\top}; \mathbf{h}_2^{(l)\top}; \ldots; \mathbf{h}_N^{(l)\top}\right]$ via using a concatenation operation "$||$", where K represents the number of heads. Here, α_{12}, α_{13}, and α_{14} represent the learned attention coefficients between node $\mathbf{h}_1$ and $\mathbf{h}_2$, $\mathbf{h}_3$, $\mathbf{h}_4$, respectively. In addition, the red arrow represents the feature aggregation operation; (c) is the output layer to get prediction $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_1; \hat{\mathbf{y}}_2; \ldots; \hat{\mathbf{y}}_N]$ by using an average operation "*Avg*". Finally, the MSE is used to penalize the average squared differences between the output $\hat{\mathbf{y}}$ and the ground truth value $\mathbf{y}$.

Monti *et al.* [40] propose a spatial-domain model on the non-Euclidean domain which designs a universe patch operator to integrate the signals within node neighborhoods. Hamilton *et al.* [23] propose an aggregation-based inductive representation learning model, named GraphSAGE. GraphSAGE generates embeddings by sampling and aggregating features from the local neighborhood of nodes and operates by sampling a fixed-size set of neighbors rather than the full set of neighbors. In addition, Velickovic *et al.* [15] inject attention mechanism into graph learning, and propose a GAT, which aggregates node information by using an attention mechanism on graph neighborhoods. Although spatial-based methods have addressed the problem of unseen graph, edge features cannot be adequately incorporated by them for representation learning.

III. The Proposed Method

This section details our proposed EAGAT model (see Fig. 1). First, we briefly introduce the methodology of GAT consisting of the attention mechanism and the node aggregation process. Next, we detail the proposed EAGAT. Finally, we exhibit the whole procedure of our proposed algorithm.

A. Graph Attention Network

The attention mechanism has been successfully used in many tasks such as machine translation [41], machine reading [42], and so on. Recently, GAT, which performs graph representation learning via using the attention mechanism has also been proposed and obtained satisfactory performance. Considering the merits of attention mechanism in characterizing pairwise importance among graph nodes, we employ GAT as the backbone of our method. In order to measure the importance of various neighbors, the attention coefficients for a node pair (i, j) can be acquired as

$$\alpha_{ij} = \frac{\exp(\text{LeakyRelu}(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyRelu}(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_k]))}, \tag{2}$$

where α_{ij} is the attention coefficient between node i and node j, $\mathcal{N}_i$ represents the neighbors of node i in the graph, and '$||$' is the concatenation operation. The set of input node features is $\mathbf{H} = [\mathbf{h}_1^\top; \mathbf{h}_2^\top; \ldots; \mathbf{h}_N^\top]$ stacked by $\mathbf{h}_1^\top, \mathbf{h}_2^\top, \ldots, \mathbf{h}_N^\top$ in row, where $\mathbf{h}_i \in \mathbb{R}^F$, $i = 1, 2, \ldots, N$, N denotes the number of nodes and F denotes the number of features of each node. Here, $\mathbf{W} \in \mathbb{R}^{F' \times F}$ indicates the weight matrix of a shared linear transformation which is applied to every node, $\mathbf{a} \in \mathbb{R}^{2F'}$ is the weight vector of a single-layer feed forward neural network, where F' represents the dimensionality of the updated features. In Eq. (2), α is normalized by a softmax function before the LeakyReLU [43] non-linearity operation.

Once obtained, the attention coefficients α_{ij} can be used to update the feature representations. Then new features $\mathbf{H}' = \left[\mathbf{h}_1'^\top; \mathbf{h}_2'^\top; \ldots; \mathbf{h}_N'^\top\right]$ can be obtained by

$$\mathbf{h}_i' = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j) \tag{3}$$

with a non-linearity σ (namely Exponential Linear Unit (ELU) [44]). Similar to the practice in [41], the multi-head attention is utilized here to incorporate different types of information and stabilize the learning process. To be specific, K independent attention mechanisms are applied to compute the hidden states before a concatenation operation, which results in the following output representations:

$$\mathbf{h}_i' = \overset{K}{\underset{k=1}{||}} \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j). \tag{4}$$

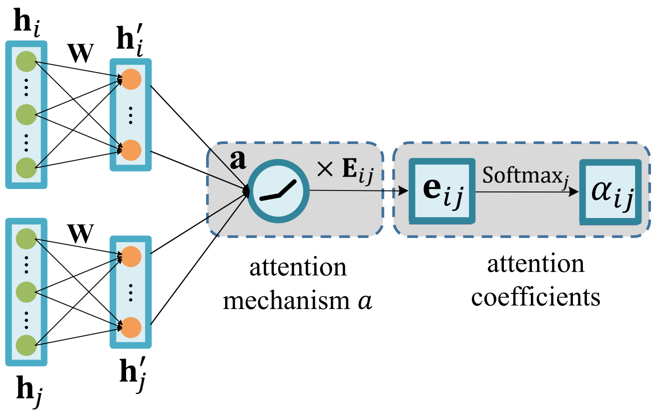

Fig. 2. The process of getting α_{ij}. The weight matrix $\mathbf{W}$ is used for feature transformation, by which $\mathbf{h}_i^{'}$ and $\mathbf{h}_j^{'}$ can be obtained. After incorporating the transformed features with the shared attention mechanism a, which is parameterized by $\mathbf{a}$ before applying the non-linearity function, the edge features $\mathbf{E}_{ij}$ is additionally utilized to generate the attention coefficients $\mathbf{e}_{ij}$. Finally, we apply the softmax function to obtain the normalized attention coefficients α_{ij}.

However, concatenation is no longer sensible when we perform multi-head attention on the node features. To deal with this issue, we employ an averaging strategy to obtain the final prediction:

$$\mathbf{h}_i^{'} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{k}\mathbf{W}^{k}\mathbf{h}_j\right), \tag{5}$$

where α_{ij}^{k} is normalized attention coefficient computed by the k^{th} attention mechanism.

The attention architecture has several interesting advantages: (1) The operation is efficient since computation can be conducted in a parallelizable way across node-neighbor pairs; (2) By assigning arbitrary weights to neighbors, the attention architecture can be applied to graph nodes of different degrees; (3) The attention architecture can be easily applied to predict previously unseen data.

B. *The Proposed EAGAT*

Traditional GAT is able to exploit the graph information, including node connectivity and features, to infer the pairwise relationships among graph nodes. However, in some cases where edge features, such as the CIO in the mobile networks, can be accessible, therefore the original form of GAT cannot make full use of the given information. Specifically, in traditional GAT, the attention coefficients are calculated only based on the node features, and thereby disregarding the edge information. To address this deficiency, the proposed EAGAT focuses on incorporating both the edge features and the node features to further improve the representation learning process (see Fig. 2).

Actually, according to the practical experience of engineers, CIO is capable to characterize the relationship between cells. Therefore, based on the conventional GAT, which is able to learn the attention coefficients between cells via exploiting their features, we regard CIO as the edge feature that can be employed to generate attention coefficients more precisely. To be specific, we denote CIO as $\mathbf{E} \in \mathbb{R}^{N\times N}$ where each element $\mathbf{E}_{ij}$ indicates the strength of connection between cell i and cell j. If there is a connection between cell i and cell j, then $\mathbf{E}_{ij}$ is equal to the value of CIO otherwise zero. After that, analogous to GAT, a weight matrix $\mathbf{W}$ is applied to embed the node features into a suitable subspace. We then perform self-attention mechanism to compute the attention coefficient between node i and node j as follows:

$$\mathbf{e}_{ij} = a(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j) \times \mathbf{E}_{ij}, \tag{6}$$

where $a : \mathbb{R}^{F^{'}} \times \mathbb{R}^{F^{'}} \rightarrow \mathbb{R}$ is a shared attentional mechanism and $j \in \mathcal{N}_i$. Eq. (6) indicates the importance between node i and node j, which is determined by both their features and edge features. To make coefficients comparable across different nodes, softmax function is used to normalize them across all choices of j:

$$\alpha_{ij} = \text{softmax}_j(\mathbf{e}_{ij}) = \frac{\exp(\mathbf{e}_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(\mathbf{e}_{ik})}. \tag{7}$$

Then the coefficients can be acquired as

$$\alpha_{ij} = \frac{\exp(\text{LeakyRelu}(\mathbf{a}^{\top}[\mathbf{W}\mathbf{h}_i||\mathbf{W}\mathbf{h}_j]) \times \mathbf{E}_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyRelu}(\mathbf{a}^{\top}[\mathbf{W}\mathbf{h}_i||\mathbf{W}\mathbf{h}_k]) \times \mathbf{E}_{ik})}. \tag{8}$$

After that, we construct the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ as

$$\mathbf{A}_{ij} = \begin{cases} \alpha_{ij} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}. \tag{9}$$

By stacking the improved graph attention layer, a multi-layer EAGAT can be achieved. The output of the l^{th} attention layer can then be obtained by

$$\mathbf{H}^{(l)} = \sigma(\mathbf{A}^{(l)}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}), \tag{10}$$

where $\mathbf{A}^{(l)}$, $\mathbf{H}^{(l)}$, and $\mathbf{W}^{(l)}$ denote adjacency matrix, the output matrix, and the trainable weight matrix of the l^{th} layer. Note that $\mathbf{H}^{(0)}$ is the input features $\mathbf{H}$, $\mathbf{A}^{(0)}$ indicates the adjacency matrix formed by the initial adjacency relationship. By applying multi-head attention, the output representations can be expressed as

$$\mathbf{H}^{(l)} = \mathop{\Big\|}_{k=1}^{K} \sigma\left(\left[\mathbf{A}^{(l)}\right]^{k}\mathbf{H}^{(l-1)}\left[\mathbf{W}^{(l)}\right]^{k}\right), \tag{11}$$

where $\left[\mathbf{A}^{(l)}\right]^{k}$, $\left[\mathbf{W}^{(l)}\right]^{k}$ denote the adjacency matrix and the trainable weight matrix in the l^{th} layer of the k^{th} attention mechanism. Therefore, final prediction can be formulated as

$$\hat{\mathbf{y}} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\left[\mathbf{A}^{(L)}\right]^{k}\mathbf{H}^{(L-1)}\left[\mathbf{W}^{(L)}\right]^{k}\right), \tag{12}$$

where L denotes the number of layers, and $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N]$ indicates the output vector of EAGAT. The convolution process of EAGAT is summarized in Algorithm 1. In our model, the MSE is adopted to penalize the differences

Algorithm 1 Edge-Aware Graph Convolution Process of EA-GAT

```
Input: Input node features $\mathbf{H}$; Input edge features $\mathbf{E}$; Neighborhood $\mathcal{N}$;
 1: // Calculate the attention coefficients
 2: for $i = 1, 2, \dots, N$ do
 3:     for $j = 1, 2, \dots, N$ do
 4:         Obtain $\alpha_{ij}$ according to Eq. (8);
 5:         Obtain $\mathbf{A}_{ij}$ according to Eq. (9);
 6:     end for
 7: end for
 8: // Perform graph convolution
 9: for $l = 1, 2, \dots, L-1$ do
10:     Obtain $\mathbf{H}^{(l)}$ according to Eq. (11);
11: end for
12: Calculate the network output according to Eq. (12);
Output: Network output $\hat{\mathbf{y}}$.
```

Algorithm 2 Proposed EAGAT for REU Estimating

```
Input: Input Graph with $\mathbf{H}$ and $\mathbf{E}$; Neighborhood $\mathcal{N}$; number of iterations $T$; learning rate $\eta$; number of graph convolutional layers $L$; number of attention heads $K$;
 1: // Train the model
 2: for $i = 1, 2, \dots, T$ do
 3:     Conduct EAGAT by Algorithm 1;
 4:     Calculate the error term according to Eq. (13), and update the weight matrices $\mathbf{W}^{k}_{(l)} (1 \leq l \geq L, 1 \leq k \geq K)$ using full-batch gradient descent;
 5: end for
 6: Conduct prediction by Algorithm 1;
Output: REU prediction for each node in the graph.
```

between the network outputs and the ground truth values $\mathbf{y} = [\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_N]$, which constitues the loss function:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\mathrm{y}_i - \hat{\mathrm{y}}_i)^2, \tag{13}$$

where y_i represents the ground truth value of the i^{th} node. Similar to [9], the network parameters here are learned by using full-batch gradient descent, where all nodes are utilized to perform gradient descent in each iteration. After the training process is completed, a new graph can be adopted for prediction based on the well-trained EAGAT model. Specifically, a corresponding adjacency matrix $\mathbf{A}$ will be first computed based on the attention coefficients, and then successive graph convolutions can be conducted to obtain the network output $\hat{\mathbf{y}}$. The implementation details of our proposed method are shown in Algorithm 2.

IV. Experiments

In this section, we conduct exhaustive experiments to evaluate the proposed EAGAT method, and also provide the corresponding algorithm analyses. To be specific, we compare EAGAT with other approaches on two real-world mobile network datasets. By following [1], R-squared (R^2) score [45] is adopted to evaluate the effectiveness of the regression model. Moreover, we demonstrate that the participation of the edge features in our EAGAT is beneficial to obtain improved performance.

TABLE I
Characteristics of two mobile network datasets.

Datasets	$Mobile\_Spring$	$Mobile\_Summer$
# Graphs	62	65
# Nodes	1266	1266
# Average Edges	16677	16245

A. Datasets

In our experiment, we use two datasets which are named $Mobile\_Spring$ and $Mobile\_Summer$. Both datasets are collected from the base stations of the metropolitan mobile network in different time periods (i.e. spring and summer) in a city of China. We regard each cell as a graph node and determine the adjacency relationship according to whether a switching occurs between the cells. Meanwhile, REU is regarded as the target value of each cell. In addition to these common information, edge features ($e.g.$, CIO, etc) are provided among the cells. Note that each dataset contains a different number of graphs, where the graphs contain the same number of nodes. The characteristics of these two datasets are summarized in Table I.

B. Experimental Setup

In our experiment, our algorithms are implemented in Python on the TensorFlow platform [46] with Adam optimizer [47]. We apply a four-layer EAGAT model with the number of attention heads being six. Here, the optimal number of attention heads is selected from the set $\{2, 4, 6, 8\}$. During the training phase, we apply an L_2 regularization in order to alleviate the over-fitting problem. Analogous to the GAT model, the ELU [44] is employed as the non-linearity across all layers.

To evaluate the performance of our proposed method, several regression models are used for comparison. Specifically, we employ two classic regression models, i.e., Multi-Layer Perceptron (MLP) [48] and Random Forest (RF) [49], together with two GCN-based methods, i.e., GAT [15] and GraphSAGE [23]. Meanwhile, we also compare the proposed EAGAT with the GCN model that can handle edge features, namely EGNN [29]. The baseline methods are implemented with the parameter setup suggested in their respective literature. Concretely, We use MLP with a single hidden layer containing 64 nodes, where the number of hidden layers is selected from the set $\{1, 2, 3, 4\}$. In RF, we set the number of trees in the forest to 100, which is selected from the set $\{10, 50, 100, 200\}$, and the maximum depth of the tree is selected from the set $\{10, 20, 30, 40\}$. In GraphSAGE, the mean aggregator is used for generating the representation of each node. In EGNN, we employ attention-based EGNN layer. Furthermore, We perform five-fold cross-validation to record the mean R^2 score and standard deviations for all the compared algorithms. Some other explanations for baseline methods are given as follows:

- In MLP and RF, only node features are used as the input while ignoring the graph structure.
- In GAT and GraphSAGE, the structures of the graph are utilized without any edge information.
- In EGNN, we use both the graph structure and the edge features.

C. Evaluation of Regression Models

For each dataset, the data is split into the training set and the test set respectively. We use the training data to learn the regression model and then test it on the test set. The effectiveness of regression model is measured by the R^2 score. As introduced in [45], R^2 score is a reliable statistical measure to show how close the data points are to the fitted regression curve, and it has been used to evaluate various regression models such as [50], [51]. Specifically, for a graph g, assume that the ground truth values of the nodes are $\mathbf{y} = [\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n]$, and the predictions are $\hat{\mathbf{y}} = [\hat{\mathrm{y}}_1, \hat{\mathrm{y}}_2, \ldots, \hat{\mathrm{y}}_n]$, where n is the number of nodes in the graph. The R^2 score for graph g can be express as

$$R_g^2 = 1 - \sum_{i=1}^{n} \frac{(\mathrm{y}_i - \hat{\mathrm{y}}_i)^2}{(\mathrm{y}_i - \bar{\mathrm{y}})^2}, \tag{14}$$

where $\bar{\mathrm{y}} = \frac{1}{n}\sum_i \mathrm{y}_i$. If $R_g^2 = 1$, the model can fit the data perfectly; if $R_g^2 = 0$, the model becomes just a naive predictor which always predicts by random; if the score is negative, the model is even worse than the naive predictor. We use the average R^2 score as a measure of the effectiveness of the regression models, i.e., $\overline{R}^2 = \frac{1}{M}\sum_{g\in G} R_g^2$, where G is the graph set, and M denotes the number of graphs.

D. Regression Results

To show the effectiveness of our proposed EAGAT, here we quantitatively and qualitatively evaluate the regression performance by comparing EAGAT with the aforementioned baseline methods.

The quantitative results obtained by different methods on the two mobile network datasets are summarized in Table II, where the highest value in each dataset is highlighted in bold. We observe that the classic methods including MLP and RF achieve relatively low accuracy, which is due to the reason that they can only utilize the features of nodes, so the graph structure cannot be captured. By contrast, GCN-based methods such as GAT and GraphSAGE are capable of integrating the information of graph structure, so they can yield better performance than MLP and RF. The EGNN algorithm, which combines graph structure and edge information, ranks in second place. This implies that the edge features are quite useful

TABLE II
R^2 SCORE (MEAN $\pm$ STD) OF EVERY COMPARED APPROACH ON THE ADOPTED DATASETS.

Model	*Mobile_Spring*	*Mobile_Summer*
MLP [48]	0.602 $\pm$ 0.007	0.614 $\pm$ 0.006
RF [49]	0.610 $\pm$ 0.008	0.618 $\pm$ 0.007
GraphSAGE [23]	0.702 $\pm$ 0.010	0.712 $\pm$ 0.011
GAT [15]	0.721 $\pm$ 0.007	0.734 $\pm$ 0.013
EGNN [29]	0.733 $\pm$ 0.008	0.745 $\pm$ 0.009
EAGAT	**0.880 $\pm$ 0.010**	**0.891 $\pm$ 0.008**

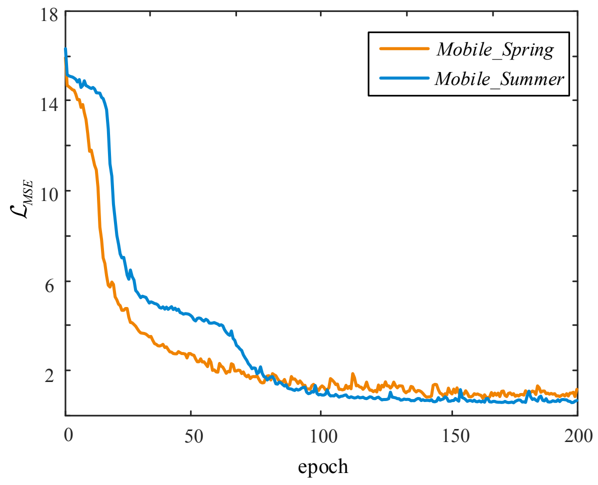

Fig. 3. The convergence curves of the model on two mobile network datasets during the training process.

to enhance regression performance. Furthermore, we observe that the proposed EAGAT achieves the top-level performance among all the methods, and the standard deviations are also very small, which reflects that the proposed EAGAT is more effective and stable than the compared methods in our REU estimation problem.

The convergence curves of our model on two mobile network datasets during the training process are shown in Fig. 3, and one can observe that the proposed EAGAT actually can converge after about 100 iterations, which demonstrates the efficiency of our model. We also compare distributions of the REU outputs of our model with the ground truth values on the two mobile network datasets. The results are given in Fig. 4, which demonstrates that the distribution of our outputs is close to that of the ground truth values, thus again reflecting the effectiveness of our model.

E. Ablation Study

The superiority of the proposed EAGAT approach has been verified by the experimental results presented above. In this section, we conduct an ablation study to further demonstrate the effectiveness of EAGAT. Different from conventional GAT which only incorporates node features to implement the self-

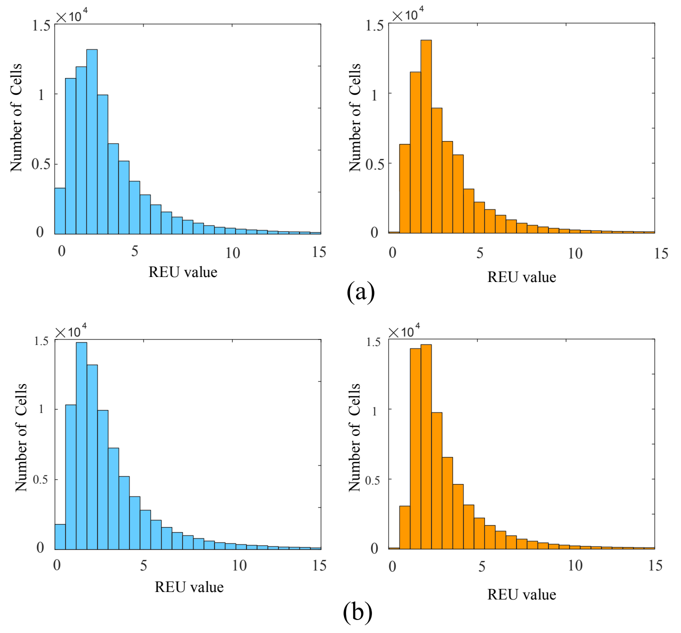

Fig. 4. Distributions of the ground truth values and the outputs. (a) Left: the distribution of the ground truth values on the $Mobile\_Spring$ dataset; Right: the distribution of the outputs on the $Mobile\_Spring$ dataset. (b) Left: the distribution of the ground truth values on the $Mobile\_Summer$ dataset; Right: the distribution of the outputs on the $Mobile\_Summer$ dataset.

attention mechanism, our methodology is to attend over both node features and edge features to obtain the attention coefficients. Therefore, we compare the results achieved by the GAT model with the EAGAT model on the $Mobile\_Summer$ dataset to reveal the contribution of the edge features. Note that our EAGAT degenerates to GAT if the edge features are not considered. From Table II, we can observe that there is a noticeable gap between GAT and the proposed EAGAT methods regarding R^2 score when conducting regression tasks. Fig. 5 shows the variation of R^2 score between EAGAT and GAT during the training process, from which we can observe the gap between these two methods intuitively. To be specific, the R^2 score will decrease when the edge features are removed, and thus the effectiveness and indispensability of the use of edge features are verified.

V. Conclusion

Estimating the Ratio of Edge-Users (REU) according to the properties of different cells and the load switching among them is an important issue in mobile networks, as it helps balance the load of different cells. In this paper, the recently proposed GAT is deployed to formulate REU estimation as a node regression problem. Based on the GAT backbone, we exhaustively exploit the inherited graph structure of mobile networks, where the node features (namely, the properties of different cells) and edge features (namely, the load switching actions) are fused in a principled way, in order to acquire improved predictions. To the best of our knowledge, it is the first time to automatically predict the REU by deploying the graph-based machine learning technique. The experimental

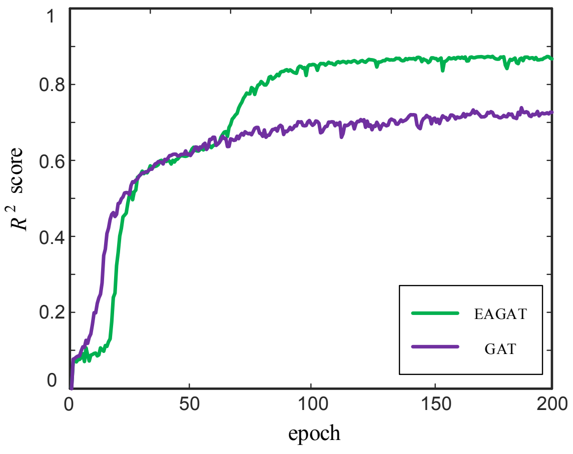

Fig. 5. R^2 score of GAT model and the proposed EAGAT model. The purple curve indicates the R^2 score of GAT. The green curve denotes the R^2 score of the proposed method EAGAT.

results on two real-world mobile network datasets demonstrate the superiority of our EAGAT approach to several state-of-the-art methods.

Acknowledgment

This research is supported by NSF of China (Nos: 61973162, 61806125, 61802247, and 61977046), the Fundamental Research Funds for the Central Universities (No: 30920032202), and the "Young Elite Scientists Sponsorship Program" by CAST (No: 2018QNRC001).

References

[1] J. Chuai, Z. Chen, G. Liu, X. Guo, X. Wang, X. Liu, C. Zhu, and F. Shen, "A collaborative learning based approach for parameter configuration of cellular networks," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1396–1404.

[2] Z. Guohua, P. Legg, and G. Hui, "A network controlled handover mechanism and its optimization in lte heterogeneous networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*. IEEE, 2013, pp. 1915–1919.

[3] A. S. Priyadharshini and P. Bhuvaneswari, "A study on handover parameter optimization in lte-a networks," in *Proceedings of the International Conference on Microelectronics, Computing and Communications*. IEEE, 2016, pp. 1–5.

[4] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the International Conference on World Wide Web*, 2013, pp. 37–48.

[5] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[7] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[8] R. S. Michalski, "A theory and methodology of inductive learning," in *Machine Learning*. Springer, 1983, pp. 83–134.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017.

[10] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6857–6866.

[11] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. Frank Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1576–1585.

[12] S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang, "Multiscale dynamic graph convolutional network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 5, pp. 3162–3177, 2019.

[13] S. Wan, C. Gong, P. Zhong, S. Pan, G. Li, and J. Yang, "Hyperspectral image classification with context-aware dynamic graph convolutional network," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–16, 2020.

[14] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, 2018.

[16] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proceedings of the International Conference on World Wide Web*, 2018, pp. 1063–1072.

[17] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[18] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[19] Z. Gao, G. Fu, C. Ouyang, S. Tsutsui, X. Liu, and Y. Ding, "edge2vec: Learning node representation using edge semantics," *arXiv preprint arXiv:1809.02269*, 2019.

[20] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[21] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3656–3663.

[22] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multi-graph convolutional networks," in *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2018, pp. 397–400.

[23] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[24] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2019.

[25] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[26] S. S. Mwanje and A. Mitschele-Thiel, "A q-learning strategy for lte mobility load balancing," in *Proceedings of the IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*. IEEE, 2013, pp. 2154–2158.

[27] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[28] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[29] L. Gong and Q. Cheng, "Exploiting edge features for graph neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9211–9219.

[30] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[31] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[32] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[33] C. Gong, J. Yang, and D. Tao, "Multi-modal curriculum learning over graphs," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 4, pp. 1–25, 2019.

[34] C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, and J. Yang, "Deformed graph laplacian for semisupervised learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2261–2274, 2015.

[35] C. Zhang, D. Ren, T. Liu, J. Yang, and C. Gong, "Positive and unlabeled learning with label disambiguation." in *Proceedings of International Joint Conference on Artificial Intelligence*, 2019, pp. 4250–4256.

[36] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[37] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[38] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proceedings of the Advances in Neural Information Processing Systems*, 2015, pp. 2224–2232.

[39] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 2014–2023.

[40] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the Advances In Neural Information Processing Systems*, 2017, pp. 5998–6008.

[42] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 551–561.

[43] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[44] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *Proceedings of the International Conference on Learning Representations*, 2016.

[45] A. C. Cameron and F. A. Windmeijer, "An r-squared measure of goodness of fit for some common nonlinear regression models," *Journal of Econometrics*, vol. 77, no. 2, pp. 329–342, 1997.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of USENIX Conference on Operating Systems design and Implementation, OSDI'16*, 2016, pp. 265–283.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.

[48] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)-a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[49] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[50] A. Gelman, B. Goodrich, J. Gabry, and A. Vehtari, "R-squared for bayesian regression models," *The American Statistician*, vol. 73, no. 3, pp. 307–309, 2019.

[51] M. A. Bujang, N. Sa'at, T. M. I. T. A. Bakar *et al.*, "Determination of minimum sample size requirement for multiple linear regression and analysis of covariance based on experimental and non-experimental studies," *Epidemiology, Biostatistics and Public Health*, vol. 14, no. 3, 2017.