


# Enhancing nano grid connectivity through the AI-based cloud computing platform and integrating recommender systems with deep learning architectures for link prediction

Nagaraju Sonti<sup>1</sup> , Rukmini M. S. S.<sup>1</sup>, and Venkatappa Reddy P.<sup>2</sup>

<sup>1</sup> Department of ECE, Vignan's Foundation for Science, Technology & Research, Vadlamudi, Andhra Pradesh, India

<sup>2</sup> Azista Industries Pvt Ltd, Advanced Pixel Research Intelligence Lab, Hyderabad, Telangana, India

**Abstract.** Cloud computing has become ubiquitous in modern society, facilitating various applications ranging from essential services to online entertainment. To ensure that quality of service (QoS) standards are met, cloud frameworks must be capable of adapting to the changing demands of users, reflecting the societal trend of collaboration and dependence on automated processing systems. This research introduces an innovative approach for link prediction and user cloud recommendation, leveraging nano-grid applications and deep learning techniques within a cloud computing framework. Heuristic graph convolutional networks predict data transmission links in cloud networks. The trust-based hybrid decision matrix algorithm is then employed to schedule links based on user recommendations. The proposed model and several baselines are evaluated using real-world networks and synthetic data sets. The experimental analysis includes QoS, mean average precision, root mean square error, precision, normalized square error, and sensitivity metrics. The proposed technique achieves QoS of 73%, mean average precision of 59%, root mean square error of 73%, precision of 76%, normalized square error of 86%, and sensitivity of 93%. The findings suggest that integrating nano-grid and deep learning techniques can effectively enhance the QoS of cloud computing frameworks.

**Keywords:** cloud computing; link prediction; cloud recommendation; nano-grid application; deep learning.

## 1. INTRODUCTION

Recently, link prediction (LP) has attracted much interest due to its practical applications in real-world scenarios like friend recommendations, e-commerce, and finding potential partners. Predicting future links that will or will not occur is an LP issue. Although LP has been studied for more than 20 years, David LibenNowell and Jon Kleinberg's work considerably influenced this field and is currently gaining more attention [1]. Common neighbours (CN), resource allocation (RA), and Adamic-Adar (AA) are some of the traditional heuristic methods used in link prediction. On the other hand, supervised learning techniques like Naive Bayes, SVM, and bagging are also employed [2]. Despite numerous sophisticated LP methods, simple heuristic approaches or combinations often yield more accurate results for certain network types. The effectiveness of a given heuristic method depends on the network topology, which may differ between social networks (SNs).

This variability limits the performance of heuristic approaches. Consequently, determining the optimal heuristic strategy for a given SN often requires a trial-and-error process.

Based on the surrounding subgraph, the Weisfeiler-Lehman neural machine (WLNLM) method [3] suggested an automatic

way to recognize appropriate ways. WLNLM is regarded as an advanced link prediction method because of its high accuracy.

Link prediction is crucial in helping us understand individual connections and interactions on networking platforms. The annual growth rate of users of social networks has been consistent. With an estimated 3.9 billion people using the internet as of April 2020 [4], researchers are interested in exploring new avenues for link prediction across massive social media platforms. Forecasting links in large-scale social networks has been the subject of several efforts [5]. The Spark framework has been effectively employed in distributed computing environments for link prediction studies, enabling precise prediction of vast social networks. With numerous computing resources, link prediction analysis is now possible in less time thanks to Spark scalability features, memory computation, and parallel job processing capabilities. Spark provides a range of application properties that allow you to tailor the computation method [6].

## 2. RELATED WORK

The prediction of links has been the subject of much research and has been approached from many angles. Various heuristic methods have typically been proposed to determine the scores for every node pair. These techniques rely on structural data regarding the pair of nodes under consideration, such as their shortest path and overlapped neighbours. Preferential attachment and familiar neighbors are two metrics used to gather data

\*e-mail: 181PG05201@vignan.ac.in

Manuscript submitted 2023-08-01, revised 2024-03-05, initially accepted for publication 2024-03-26, published in July 2024.

regarding one-hop neighbors and determine these scores [7]. In addition, it has been suggested to include data about connections extending more than one hop using higher-level heuristic approaches like SimRank, PageRank, and Katz, as well as second-order heuristic strategies such as resource allocation and Adamic-Adar. These heuristic techniques are highly efficient for link prediction. Most heuristic techniques rely on manually designed structural information, which can limit their applicability. To overcome this drawback, embedding-based techniques have been recommended [8]. These approaches use the connections between nodes to learn node embeddings, which are then used to calculate similarity scores. Matrix factorization is commonly used to learn node embeddings by breaking down the graph adjacency matrix. Other techniques like Deepwalk and node2vec use random walks to generate Skip-Gram embeddings [9]. LINK [10] learns to categorize the presence of links based on the connectivity information in each row of the adjacency matrix. However, embedding methods can be brutal to generalize due to their performance being influenced by the sparsity of the input graph.

Recently, there have been attempts to use graph neural networks (GNNs) for link prediction, as they are effective at learning graph representations. GAE and VGAE [11] use GCNs to learn node representations in an auto-encoder architecture to recreate the input graph. Link prediction has seen various GNN architectures, many of which are based on the GAE. However, SEAL [12] takes a different approach by reformulating the link prediction task to include subgraph classification. Rather than directly predicting links, SEAL performs the task of graph classification. To do this, it samples enclosing graphs around every target link to compose a dataset. [13] proposed several measures based on the structural data of nodes for link prediction. Common neighbours (CNs) are one of the most commonly used measures. CN measures the similarity between two nodes by the number of shared neighbours, as proposed by [14]. Using CN, normalization techniques like Sorensen's index and Jaccard coefficient, as mentioned in [15], can increase link prediction accuracy.

To calculate the probability of a link between two nodes, a weight is given to each shared neighbour in preferential connection, and resource allocation is evaluated [16]. Recently, the paper [17] proposed a novel similarity measure depending on the tree-augmented naive (TAN) Bayes likelihood-based model. Better link predictions are produced by the TAN model because it considers the relationship among shared CN [18]. The global structural information of nodes, including paths and errands, has also been the subject of numerous studies [19]. Each pair of connected nodes has had its similarity evaluated using local path techniques based on the two- and three-hop neighbours of each other [20]. The similarity score was calculated using all paths of various lengths between nodes in [21]. At the same time, the SimRank technique was proposed, which assumes that if two nodes are connected to the same nodes, they are comparable. Notation and its explanation are shown in Table 1.

**Table 1**

Notation and its explanation

Notation	Explanation
$F(x)$	objective function
$\rho_t$	weight
$r_{ti}$	pseudo-residual representing
R1() and R2()	pseudo-random number generator (PRNG) functions
$V_{bc}$	behavioural constraint
C	set of classes
S	to calculate entropy
A	information gain
H(S)	the set S's entropy
$V_{br}$	rate of behavior change
p(c)	ratio of the number of items in class c
$w_1$	weight coefficients
$F_{t-1}^{x_i}$	negative gradient of the loss function
H	group of all potential regression trees
$V_{be}$	behavioural experience denoted

### 3. SYSTEM MODEL

This section proposes link prediction for data transmission in cloud networks using deep learning techniques based on cloud recommendation. Heuristic graph convolutional networks were used to predict data transmission links, and the trust-based hybrid decision matrix algorithm was utilized to schedule links based on user recommendations. Figure 1 illustrates the system framework for designing and predicting cloud links.

#### 3.1. Cloud network data transmission link prediction using heuristic graph convolutional networks

The goal is to assign a set of data packets, including both primary and sensitive data, to different cloud servers, minimizing execution time while ensuring encryption of sensitive data and some preliminary data. The input data packets are separated into distinct sub-packages, with details on the length of each packet and the time spent in each working mode. Cloud service providers offer encryption and non-encryption modes of operation. The output is a plan for assigning data packets to different clouds, considering execution time and security.

To achieve this, a similarity score is computed using only the structural characteristics of neighbours that coincide with the specified nodes. The structural elements of each node are used as the foundation for this operation. However, conventional GCN cannot compute this score due to a normalized adjacency matrix and hidden representation dimension being more minor than the number of nodes. The small size makes it difficult to distinguish the characteristics of each neighbourhood after aggregation, preventing GCN from detecting overlapping areas.

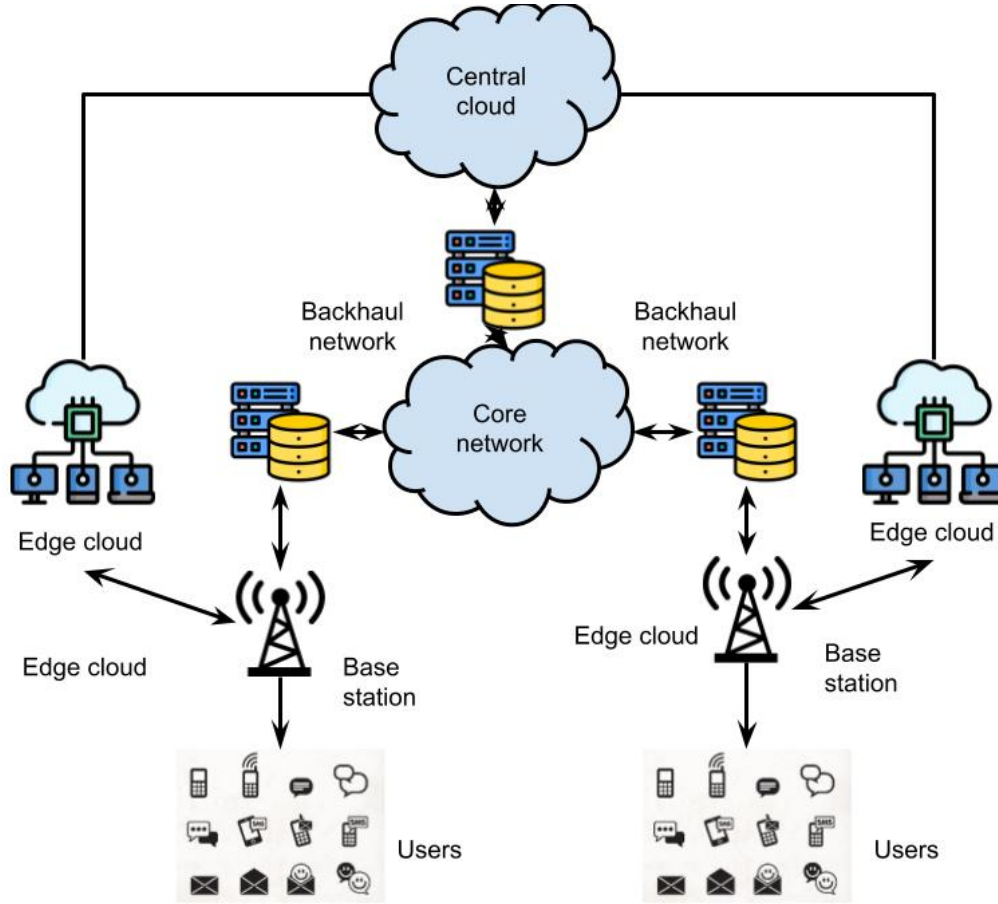


Fig. 1. Framework for cloud link prediction and scheduling

Additionally, the normalized adjacency matrix prevents GCN from counting multiple neighbourhoods. To address this, the neighbourhood overlap-aware aggregation scheme is proposed to determine the neighbourhood overlap-aware score.

The proposed system framework uses heuristic graph convolutional networks to predict data transmission links in cloud networks. The trust-based hybrid decision matrix algorithm then schedules links based on user recommendations. Figure 2 illustrates the system framework for designing and predicting cloud links.

An adjacency matrix teaches GCNs beneficial structural traits, and they estimate similarity scores based on overlapping neighbourhoods. (a) First, GCNs use the structural feature generator  $F$  to create a structural feature vector  $X_{\text{struct}} \in R^{N \times 1}$  from an adjacency matrix  $A \in R^{N \times N}$ . When only features of overlapped neighbours between nodes are to be considered, GCNs (a) build a diagonal matrix  $X_{\text{struct}} \in R^{N \times N}$  and (b) multiply the sum of powers of adjacency matrices to aggregate the features of multi-hop neighbourhoods. To compute similarity scores and adaptively mix them with the learnable parameter ( $d$ ).

The purpose of machine learning methods is to obtain an approximation,  $F(x)$ , of objective function  $F(x)$ , which maps instances  $x$  to their output values  $y$ , given a training dataset  $D = \{x_i, y_i\}_1^N$ . The learning process can generally be viewed as an optimization problem where the goal is to minimize the

anticipated value of a particular loss function,  $E[L(y, F(x))]$ . This predicted loss can be roughly estimated using the data:  $\sum_{i=1}^N L(y_i, F(x_i))$ .

The approach is constructed using the additive expansion in equation (1) in the specific case of gradient boosting

$$F_t(x) = F_{t-1}(x) + \rho_t h_t(x), \quad (1)$$

where  $\rho_t$  is the  $t$ -th function weight and  $h_t(x)$ . The approximation is built in stages, with each step creating a new model  $h_t$  without altering any existing models in  $F_{t-1}(x_i)$ . Initially, equation (2) initializes the additive model with a constant approximation

$$F_0(x) = \operatorname{argmin}_{\alpha} \sum_{i=1}^N L(y_i, \alpha) \quad (2)$$

and to reduce equation (3), the following models are created

$$(\rho_t, h_t(x)) = \operatorname{argmin}_{\rho, h} \sum_{i=1}^N L(y_i, F_{t-1}(x_i) + \rho h_t(x_i)). \quad (3)$$

Nevertheless, the problem is divided into two parts rather than jointly solving the optimum for  $h_t$ . Each method is first

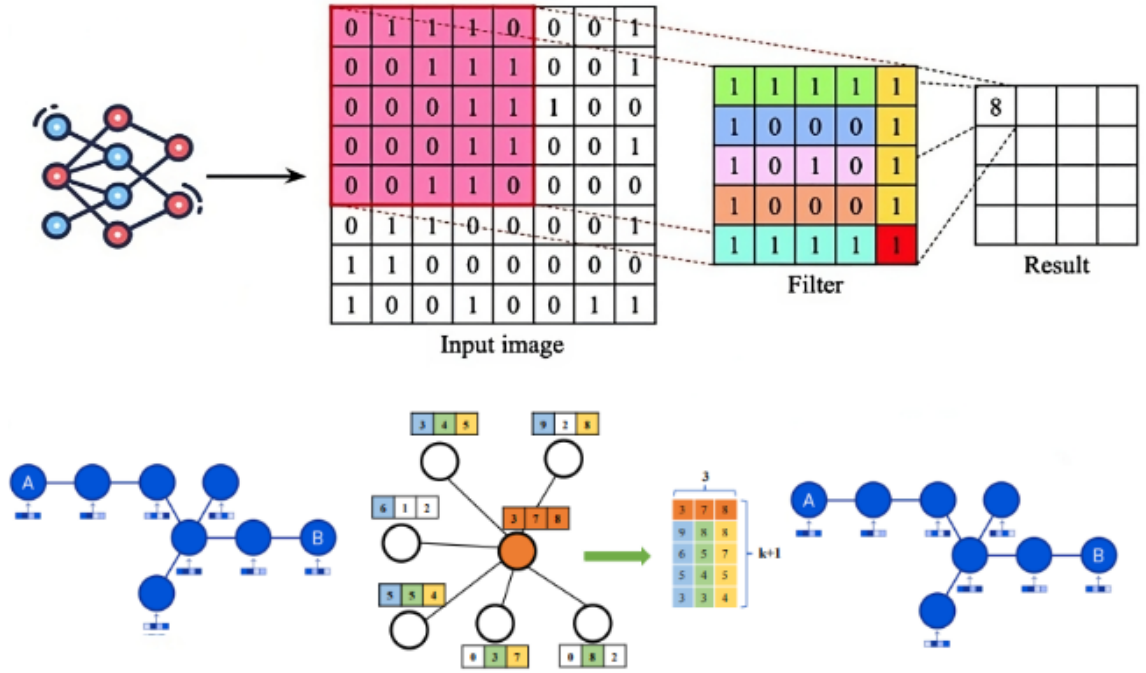


Fig. 2. The link prediction framework of the GCN

trained to discover the gradient vector of the loss-data-based function. To do this, each model,  $h_t$  is trained on a fresh dataset,  $D = \{x_i, r_{ti}\}_{i=1}^N$ , where  $r_{ti}$  is the pseudo-residual representing the loss function's negative gradient at  $F_{t-1}^{x_i}$  by equation (4)

$$r_{ti} = - \left. \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right|_{F(x)=F_{t-1}(x)}. \quad (4)$$

For the given data points, which are parallel to the gradient of  $L$  at  $F_{t-1}(x)$ , the function,  $h_t$ , is anticipated to produce values that are close to the pseudo-residuals ( $x$ ). But remember that  $h$ 's training is typically influenced by square-error loss, which may differ from the provided objective loss function. However, after solving an optimization issue using line search on the provided loss function, the value of  $t$  is determined. Lasso combines equation (11) regularisation and linear classification (5)

$$\min_w \sum_{(x_i, y_i)} l(x_i, y_i, w) + \lambda |w|_1 \quad (5)$$

the element-wise operator equation (6) definition of the capped (11) norm

$$q_c(w_i) = \min(|w_i|, \epsilon). \quad (6)$$

By modifying the associated regularisation parameters,  $\mu\lambda \geq 0$  by equations (7) and (8), one can regulate trade-off among feature extraction and regularisation when capped (11) norm is paired with a regular (11) or (12) norm

$$\min_w \sum_{(x_i, y_i)} l(x_i, y_i, w) + \lambda |w|_1 + \mu q_c(w); \quad (7)$$

$$\text{Here } q_c(w) \text{ denotes } [q_c(w_1), \dots, q_c(w_d)]. \quad (8)$$

Boosting assumes that limited-depth regression trees are utilized to pre-process data.  $H$  denotes the group of all potential regression trees. Assume  $|H|$  to be finite by taking into account the restricted precision and identifying trees that get the same values across the whole training set as belonging to the same tree (albeit possibly large). We suggest learning a linear classifier in this transformed space, considering that inputs are mapped into  $R|H|$  through  $\phi(x) = [h_1(x), \dots, h_{|H|}(x)]^T$ . Equation (9) changes to

$$\min_{\beta} \sum_{(\phi(x_i), y_i)} l(\phi(x_i), y_i, \beta) + \lambda |\beta|_1 + \mu q_c(\beta). \quad (9)$$

A sparse linear vector that selects trees in this case is  $\beta$ . Despite being very high dimensional, optimization in equation (10) is tractable because  $\beta$  is quite sparse. We derive a final classifier by assuming that trees in  $H$  are arranged such that the first  $T$  entries are non-zero  $\beta$ , without sacrificing generality

$$H(x) = \sum_{t=1}^T \beta_t h_t(x). \quad (10)$$

The revised parameters are given by equations (11)–(18):

$$w_{ij}(t+1) = w_{ij}(t) - \eta_w \Delta w_{ij}, \quad (11)$$

$$\Delta w_{ij} = \frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_{ij}}, \quad (12)$$

$$m_{ij}(t+1) = m_{ij}(t) - \eta_m \Delta m_{ij}, \quad (13)$$

$$\Delta m_{ij} = \frac{\partial L}{\partial m_{ij}} = \frac{\partial L}{\partial s_{ij}} \frac{\partial s_{ij}}{\partial m_{ij}}, \quad (14)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) - \eta_{\sigma} \Delta \sigma_{ij}, \quad (15)$$

$$\Delta\sigma_{ij} = \frac{\partial L}{\partial \sigma_{ij}} = \frac{\partial L}{\partial s_{ij}} \frac{\partial s_{ij}}{\partial \sigma_{ij}}, \quad (16)$$

$$k_{f,v}(t+1) = k_{f,v}(t) - \eta_k \Delta k_{f,v}, \quad (17)$$

$$\Delta k_{f,v} = \frac{\partial L}{\partial k_{f,v}} = \frac{\partial L}{\partial C_{fi}} \frac{\partial C_{fi}}{\partial k_{f,v}}. \quad (18)$$

### 3.2. Link recommendation using trust-based hybrid decision matrix algorithm

When evaluating the trust of a device identity, it is crucial to consider privacy issues. If a device ID is made public, it could be vulnerable to identity fraud, creation of false data, and other malicious activities. Public key infrastructure (PKI) is a secure authentication technique used in edge computing that simplifies one-to-one communication in a distributed context. Since identity trust is a critical component of overall trust evaluation, this study proposes a critical generation approach that reduces computational complexity and simplifies various PKI elements. During the resource request process, Identity Trust VI assesses the reliability of ECU identities. We assume that the elliptic curve cryptosystem (ECC)-based anonymous verification approach preserves ECU identity trust [11], which is used to generate keys for each ECU to compute VI. The following two steps involve creating the public and private keys after the ECC produces a discrete elliptic curve named  $E$ :

**Step 1:** Generating server keys. The neighbourhood edge server produces the critical pair  $(K_p, K_s)$  presented in equations (19) and (20)

$$K_p = H(I_l, I_c), \quad (19)$$

$$K_s = M(K_p). \quad (20)$$

The hash function  $H$  uses ID  $I_e$  of ECUs and ID  $I_l$  of edge server to calculate public key  $K_p$ . Encoding function  $M$  calculates the private key  $K_s$  based on  $K_p$ .

Local edge servers transmit identity information  $D = \{K_p, E, O\}$  to ECUs after calculating  $K_p$  and  $K_s$ .

**Step 2:** Key generation of ECUs. ECUs calculate the three keys  $K_{vi}$ ,  $K_{ep}$ , and  $K_{es}$  after receiving  $D$  as an equation (21) and (22):

$$K_{vi} = R_1(O(x, y)) \quad K_{ep} = R_2(O(x, y)), \quad (21)$$

$$K_{es} = R_1(M(K_{ep})). \quad (22)$$

Virtual key where  $R_1()$  and  $R_2()$  are pseudo-random number generator (PRNG) functions,  $K_{es}$  and  $K_{ep}$  are the private and public keys of ECUs, and  $K_{vi}$  is used to secure the genuine ID of ECUs. The local edge server receives a message  $F = \{IDe, K_v, K_{ep}, a\}$ ,  $a$  from the ECUs at that point. Edge servers and ECUs communicate with one another during the later authentication stage using the public key, and the private key is utilized to complete the identity matching.  $V_i$  is set to 1 if ECUs successfully authenticate; otherwise,  $V_i$  is set to 0.

The proposed framework uses fuzzy rules to construct trust values, with subjective and objective trust values serving as fuzzy inputs and fuzzy outputs as trust values. (i) Fuzzy inputs have three states: high, medium, and low, with values between

0 and 1. (ii) With values between 0 and 1, fuzzy outputs have three states: low, medium, and high. The trust repository is updated as part of the agent's responsibility to update trust values. Uncertainty, mistrust, and trust are the three states in the fuzzy rules map that produce a trust value. If the gathered trust value is low, the CSPj trust value equates to mistrust. If the CSPj trust value is high, it maps to trust; if not, it maps to doubtful.

Behavior trust

$$V_{bc} = w_1 \xi_1 + w_2 \xi_2 + \dots + w_c \xi_c. \quad (23)$$

Here, behavioural constraint is denoted by  $V_{bc}$ , behavioural experience is denoted by  $V_{be}$ , and  $V_{br}$  indicates behavioural change rate. Weight coefficients  $wbc + wbe + wbr = 1$  can be altered to suit specific jobs.

1) Behavior constraint: Various ECUs notice a varying number of interactions or behaviour limitations that are imposed by particular activities according to equation (24):

$$V_{bc} = w_1 \xi_1 + w_2 \xi_2 + \dots + w_c \xi_c. \quad (24)$$

Behavior experience: Since the environment for edge computing is constantly changing, it is vital to take change of trust into account dynamically. ECUs that will be tested are listed as ECU<sub>c</sub>. Behaviour experience  $V_{bh}$  implies interactive reliability of the recent past for interactions between several ECUs, where little engagement can nonetheless result in strong trust-building. Let us say that ECUs have been communicating with ECU<sub>1</sub>, ECU<sub>2</sub>, and ECU<sub>k</sub>, which are the other  $k$  ECUs. Given by equation (25)

$$s_j(\Delta t) = \{\tau_j^1, \tau_j^2, \dots, \tau_j^s, \dots, \tau_j^t\}. \quad (25)$$

Define by equation (26)

$$H_j^+(\Delta t) = \sum_{i=1}^t 1(\tau_j^i \geq \alpha_j). \quad (26)$$

If the condition is true, the  $1(\cdot)$  function returns 1, otherwise it returns 0. Equation (27) is utilized to define  $V_{be}$

$$V_{be} = \frac{1}{k} \sum_{j=1}^k \frac{H_j^+(\Delta t)}{t}. \quad (27)$$

Rate of behavior change: Rate of behaviour change, which is denoted as  $V_{br}$  and reflects the actual change in behaviour trust, which changes with time. By way of equation (28)

$$V_{br} = \frac{1}{k} \sum_{j=1}^k \left( \frac{(1-\lambda)V_{be\Delta(t-1)}}{1 + \sqrt{t - H_j^+(\Delta(t-1))}} + \frac{\lambda V_{be\Delta(t)}}{1 + \sqrt{H_j^+(\Delta t)}} \right). \quad (28)$$

Trusted capability: Capability trust is the weight given to a device capacity properties, such as response time, available bandwidth, accessibility, and so forth.

In this study, we propose a hybrid approach that combines a decision tree and a neural network to enhance the performance

of classification decisions. A neural network is integrated into each decision tree node to achieve improved results compared to using each technique individually. Neural networks excel at classifying items into smaller categories, while their performance improves with an increase in the number of categories. Decision trees, on the other hand, can handle multiple distinct categories by employing a collection of binary options such as 0 or 1 to construct the tree. The attribute with the best information gain is selected to obtain the same outcome. We use entropy, a widely used information theory metric that indicates the (im) purity of any set of samples, to calculate information gain. Entropy  $H(S)$  measures the degree of uncertainty in a given dataset  $S$ .

$H(S) = \sum_{c \in C} -p(c) \log_2(p(c))$ , where  $S$  is the dataset that is used to calculate entropy at this time;  $p(c)$  is the ratio of the number of items in class  $c$  to the total number of elements in set  $S$ .  $C$  is the set of classes in  $S$ , where  $C = 0, 1$ . The set  $S$  is perfectly categorized if  $H(S) = 0$ .

The difference in entropy between before and after the set  $S$  is split on a result attribute  $A$ , which is known as information gain or  $IG(A)$ . After splitting set  $S$  on outcome attribute  $A$  by equation (29), the amount of uncertainty  $S$  was reduced is represented by this quantity

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t), \quad (29)$$

where  $H(S)$  is the set  $S$  entropy;  $T$  is the subsets produced when set  $S$  was divided by the outcome attribute  $A$  so that in equation (30)

$$S = \bigcup_{t \in T} t. \quad (30)$$

The portion of items in set  $t$  divided by the total items in set  $S$  is represented as  $p(t)$ , while  $H(t)$  is the entropy of subset  $t$ . The remaining properties are evaluated for their information gain, and the property with the highest gain is selected to split set  $S$  at each iteration. Then, a neural network is built using the attribute with the highest information gain. A separate neural network is constructed for each binary classification only when the issue is present. At each decision tree node, the neural network consists of all possible outcome attributes that could result in a 0 or 1.

## 4. PERFORMANCE ANALYSIS

The effectiveness of the proposed approach is demonstrated by comparing it with standard techniques using various real benchmark dynamic networks. The experiments are conducted on a server coded in Python with 16 GB RAM and a tensor flow anaconda environment.

### 4.1. Parameter setup

The approach employed in this study involves an encoder method that combines a convolutional network and a graph network. Each dataset is segmented at predetermined intervals, resulting in 320 network sequences. The training set comprises the first 240 networks, while the remaining 80 networks form the test set. The model consists of a single convolutional network and four GCN networks, with each GCN network utilizing a Chebyshev polynomial hyperparameter  $K$  of 3. The hidden layer dimensions of LSTM for the first five datasets are 256, while the last dataset uses 512. Similarly, the hidden layer dimension of the GCN network for the first five datasets is 512, while the last dataset uses 512. The model undergoes 200 iterations during the training phase, with the inputs being eleven continuous network sequences from the training set.

### 4.2. Baseline methods

In this study, we compare the effectiveness of the proposed technique with various standard methods such as DDNE, GTRBM, ctRBM, TNE, and node2vec. Node2vec. This method uses a second-order biased random walk algorithm to explore a node neighbourhood, allowing for a balance of local and network-wide features. Node2vec is used as a starting point for predicting dynamic links.

### 4.3. Proposed analysis

Table 2 presents a parametric analysis of the proposed technique based on the number of epochs. Figure 3 illustrates the results of a parametric analysis of QoS based on the number of epochs. The proposed technique achieved a QoS of 88% after 100 epochs, 89% after 200 epochs, 91% after 300 epochs, 93% after 400 epochs, and 97% after 500 epochs, depending on the number of samples used in the neural network training.

The parametric analysis for the mean average is depicted in Fig. 4. The proposed technique achieved a mean average

**Table 2**  
Parametric analysis for the proposed technique

No. of samples	QoS	Mean average precision	Root mean square error	Precision	Normalization square error	Sensitivity
100	88	91	65	79	82	77
200	89	92	68	81	83	81
300	91	93	71	82	85	83
400	93	94	73	84	87	85
500	97	95	74	85	88	86

precision of 91% for 100 epochs, 92% for 200 epochs, 93% for 300 epochs, 94% for 400 epochs, and 95% for 500 epochs.

The parametric analysis for root mean square error based on the number of epochs is presented in Fig. 5. The proposed technique achieved a root mean square error of 65% for 100

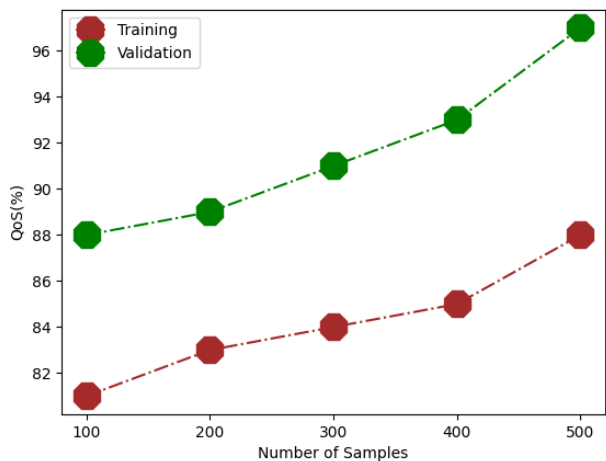


Fig. 3. Parametric analysis of QoS

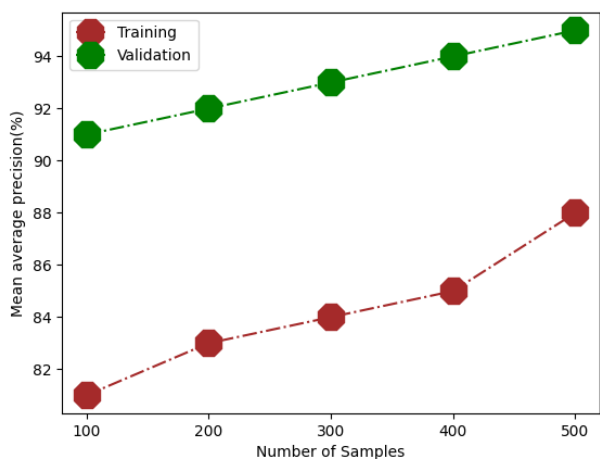


Fig. 4. Parametric analysis of mean average precision

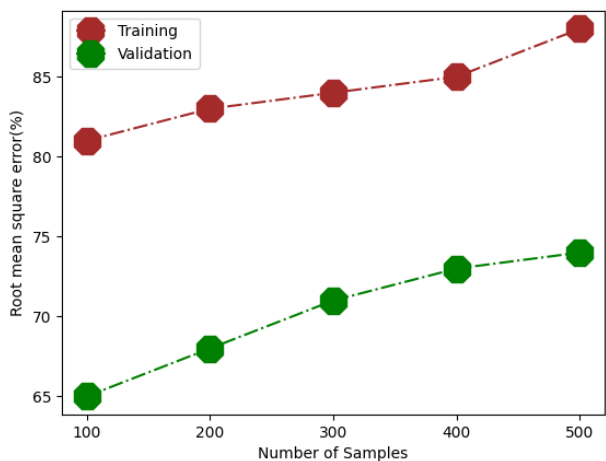


Fig. 5. Parametric analysis of root mean square error

epochs, 68% for 200 epochs, 71% for 300 epochs, 73% for 400 epochs, and 74% for 500 epochs, based on the number of samples used in the neural network training.

The results of the parametric analysis for precision based on neural network training for different numbers of samples are presented in Fig. 6. The proposed technique achieved an accuracy of 79% for 100 epochs, 81% for 200 epochs, 82% for 300 epochs, 84% for 400 epochs, and 85% for 500 epochs.

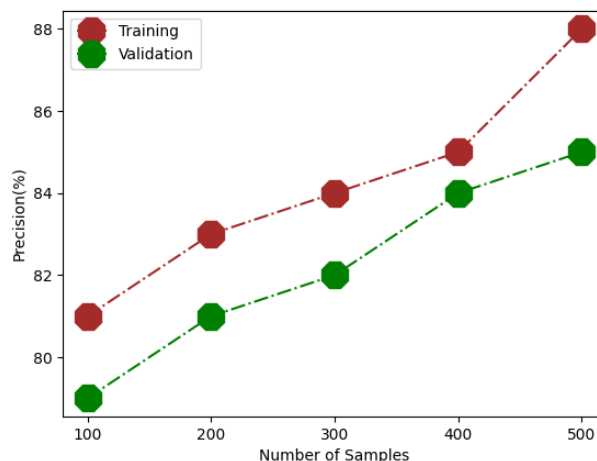


Fig. 6. Parametric analysis of precision

The parametric analysis for normalized square error based on the number of epochs is depicted in Fig. 7 above. According to the results, the proposed technique achieved a normalized square error of 82% for 100 epochs, 83% for 200 epochs, 85% for 300 epochs, 87% for 400 epochs, and 88% for 500 epochs based on the number of samples in the neural network training.

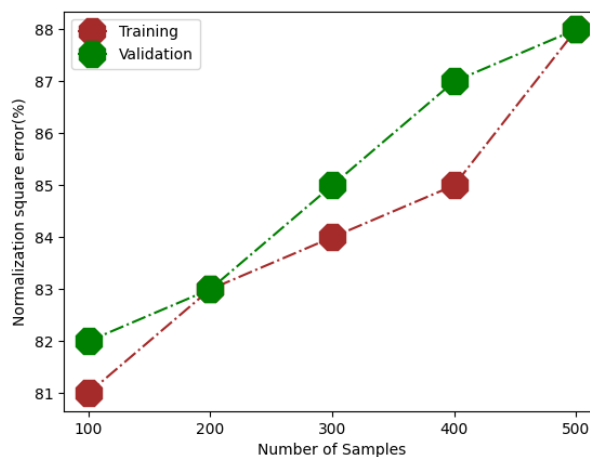


Fig. 7. Parametric analysis of normalization square error

Figure 8 displays the parametric analysis for sensitivity based on the number of samples in the neural network training. The proposed technique achieved a sensitivity of 77% for 100 epochs, 81% for 200 epochs, 83% for 300 epochs, 85% for 400 epochs, and 86% for 500 epochs.

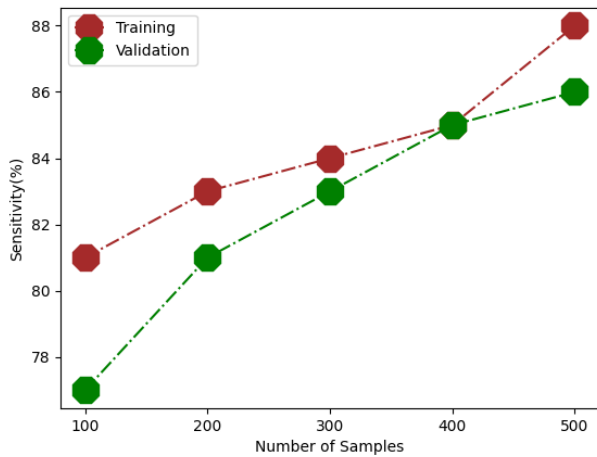


Fig. 8. Parametric analysis of sensitivity

#### 4.4. Comparative analysis

Table 3 compares the performance of the proposed technique with existing techniques such as node2vec, TNE, ctRBM, GTRBM, and DDNE. The analysis is based on several metrics, including QoS, mean average precision, root mean square error, precision, normalized square error, and sensitivity.

Figure 9 depicts the analysis of baseline techniques, including node2vec, TNE, ctRBM, GTRBM, and DDNE, for QoS. The suggested method yielded a QoS of 61%, compared to 59% for the current GCN and 59% for VGAE using the node2vec baseline approach. The proposed method obtained a QoS of 65% for the TNE baseline technique, compared to 59% for the current

GCN and 63% for VGAE. For the ctRBM baseline approach, the suggested technique achieved a QoS of 68%, compared to 61% for the current GCN and 65% for VGAE. The proposed technique obtained a QoS of 69% for the GTRBM baseline method, compared to 63% for the existing GCN and 66% for VGAE. Lastly, the suggested technique obtained a QoS of 73% for the DDNE baseline technique, compared to 65% for the current GCN and 71% for VGAE.

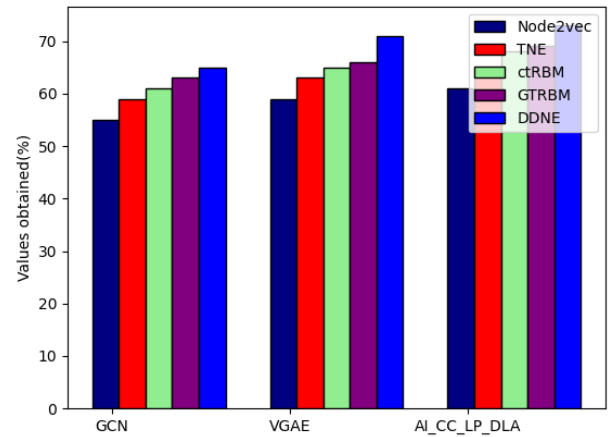


Fig. 9. Comparison of QoS

The analysis for mean average precision is shown in Fig. 10. The proposed method outperformed the current GCN with a mean average precision of 53% for the GTRBM baseline method, compared to 46% for the GCN. For the DDNE baseline

Table 3

Analysis based on various baseline methods

Dataset	Techniques	QoS	Mean average precision	Root mean square error	Precision	Normalization square error	Sensitivity
Node2vec	GCN	55	41	59	61	71	81
	VGAE	59	43	63	63	73	83
	AI_CC_LP_DLA	61	45	65	65	75	85
TNE	GCN	59	42	61	62	72	82
	VGAE	63	46	63	66	76	86
	AI_CC_LP_DLA	65	48	66	69	78	88
ctRBM	GCN	61	45	62	63	75	83
	VGAE	65	49	65	65	79	88
	AI_CC_LP_DLA	68	52	69	69	83	89
GTRBM	GCN	63	46	65	65	77	85
	VGAE	66	49	69	69	82	89
	AI_CC_LP_DLA	69	53	72	72	83	92
DDNE	GCN	65	48	68	69	79	88
	VGAE	71	55	73	74	83	92
	AI_CC_LP_DLA	73	59	75	76	86	93



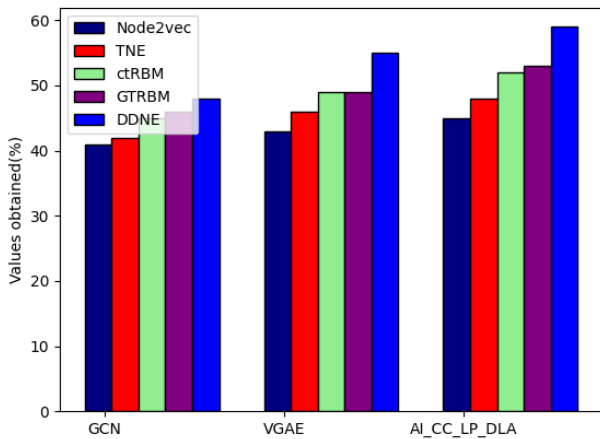


Fig. 10. Comparison of mean average precision

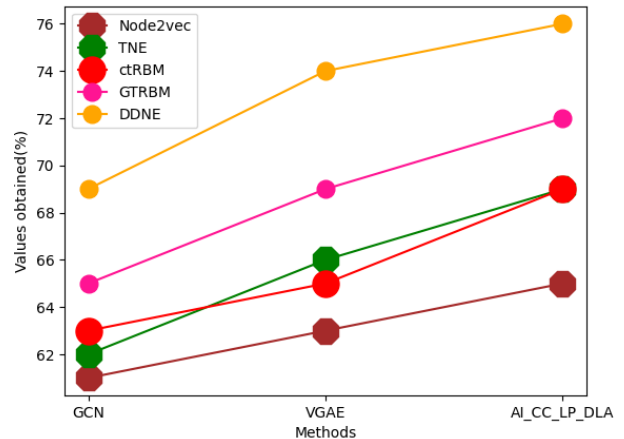


Fig. 12. Comparison of precision

method, the suggested method yielded a mean average precision of 59%, compared to 48% for the current GCN and 55% for VGAE.

The analysis for baseline methods, which include node2vec, is shown in Fig. 11. The existing GCN obtained an RMSE of 59%, whereas the suggested method achieved a value of 65%. The proposed technique received an RMSE of 66% for the TNE baseline technique, compared to 61% for the existing GCN and 63% for VGAE. The suggested method produced an RMSE of 69%, compared to 62% for the ctRBM baseline method and 65% for the existing GCN and VGAE.

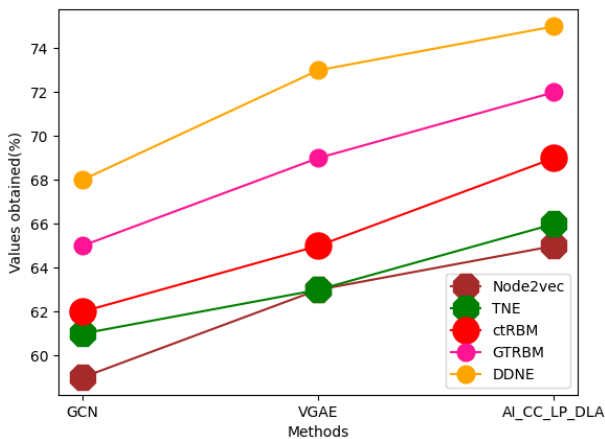


Fig. 11. Comparison of root mean square error

A precision analysis is shown in Fig. 12. The proposed method obtained a precision of 65% for the node2vec baseline method, whereas the current GCN obtained 61% and VGAE obtained 63%. The proposed technique achieved 72% precision for the GTRBM baseline technique, compared to 65% for the existing GCN and 69% for VGAE.

The analysis for normalization square error for TNE, ctRBM, DDNE, node2vec, and GTRBM baseline techniques is shown in Fig. 13. The proposed technique achieved 83% normalization square error for the GTRBM baseline method, compared to 77% for the existing GCN and 82% for VGAE. Ultimately, the

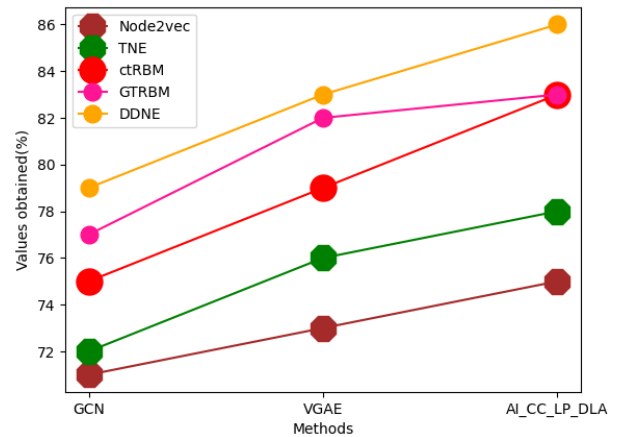


Fig. 13. Comparison of normalization square error

suggested method produced an 86% normalization square error for the DDNE baseline method.

A sensitivity analysis of the node2vec, TNE, ctRBM, GTRBM, and DDNE baseline techniques is shown in Fig. 14. For the DDNE baseline method, the proposed technique achieved a sensitivity of 93%, while existing GCN attained 88%, and VGAE attained 92%.

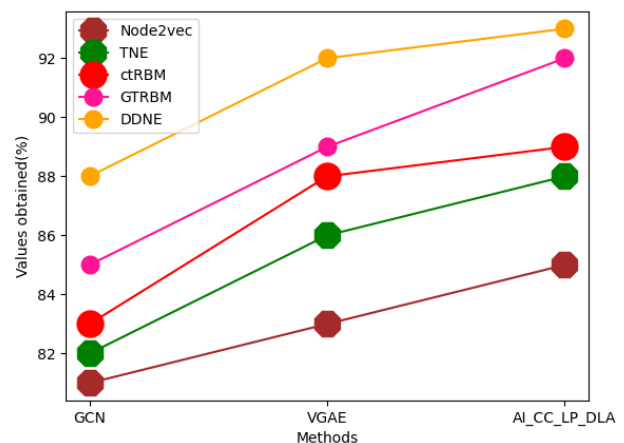


Fig. 14. Comparison of sensitivity

## 5. RESULTS OF LINK PREDICTION

For the model training, we utilized ten consecutive snapshots “Gt10, . . . , Gt1” to predict the subsequent snapshot “Gt”. There are typically two methods for processing networks: 1) Predicting the subsequent snapshot Gt utilizing only the  $(t - 1)$ -th snapshot Gt1; 2) Predicting the subsequent snapshot, Gt, by combining the previous ten snapshots (Gt10, . . . , Gt1) into a single sample. In this research, we used the first option for node2vec and the second option for other methods. Each indicator was thoroughly examined. It is observed that, for both long-term and short-term prediction capabilities, the proposed model outperforms all baseline methods, while node2vec performs worst in most cases. This implies that other dynamic prediction techniques perform significantly better because they are capable of capturing dynamic characteristics, whereas network embedding techniques, which are designed for static networks, may not be as effective in capturing temporal changes.

## 6. CONCLUSIONS

In this study, we proposed a novel link prediction method for cloud data transmission that includes secure link recommendation. Our approach leverages heuristic graph convolutional networks to predict the data transmission link and a trust-based hybrid decision matrix algorithm to recommend the user link. We achieved accurate link classification by integrating learned visualizations of related nodes to generate a vector for a link, which we then fed into a regression algorithm. We evaluated our method using three metrics and observed that it outperformed all baseline methods in terms of short- and long-term prediction capabilities. Specifically, our proposed technique achieved a QoS of 73%, mean average precision of 59%, root mean square error of 73%, precision of 76%, normalized square error of 86%, and sensitivity of 93%. These results demonstrate the potential of deep learning architectures for improving link prediction in cloud data transmission and highlight the value of combining multiple techniques to achieve better performance.

## ACKNOWLEDGEMENTS

We declare that this manuscript is original, has not been published before, and is not currently being considered for publication elsewhere.

## CONFLICTS OF INTEREST/COMPETING INTERESTS

The authors do not have any conflicts of interest.

## REFERENCES

- [1] N.N. Daud, S.H. Ab Hamid, M. Saadoon, C. Seri, Z.H.A. Hasan and N.B. Anuar, “Self-ConFig.d Framework for scalable link prediction in twitter: Towards autonomous spark framework,” *Knowledge-Based Syst.*, vol. 255, p. 109713, 2022.
- [2] J. Zheng *et al.*, “Analysis of thermal characteristics with multi-physics coupling for the feed system of a precision CNC machine tool,” *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 72, no. 2, p. e148941, 2024, doi: [10.24425/bpasts.2024.148941](https://doi.org/10.24425/bpasts.2024.148941).
- [3] M. Bohlooly Fotovat and T. Kubiak, “Non-bifurcation behavior of laminated composite plates under in-plane compression,” *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 72, no. 2, p. e148874, 2024, doi: [10.24425/bpasts.2024.148874](https://doi.org/10.24425/bpasts.2024.148874).
- [4] C. Cao, W. Dong, W. Zhang and Y. Gao, “WiEdge: Edge Computing for Audio Sensing Applications With Accurate Wireless Link Prediction,” *IEEE Internet Things J.*, vol. 10, no. 5, pp. 3982–3994, 2023, doi: [10.1109/JIOT.2022.3173668](https://doi.org/10.1109/JIOT.2022.3173668).
- [5] J. Wang *et al.*, “Diagnosis of inter-turn short circuit fault in IPMSMs based on the combined use of greedy tracking and random forest,” *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 72, no. 2, p. e148943, 2024, doi: [10.24425/bpasts.2024.148943](https://doi.org/10.24425/bpasts.2024.148943).
- [6] P. Steinbach, F. Gernhardt, M. Tanveer, S. Schmerler, and S. Starke, “Machine learning state-of-the-art with uncertainties,” *arXiv:2204.05173*, 2022, doi: [10.48550/arXiv.2204.05173](https://doi.org/10.48550/arXiv.2204.05173).
- [7] Y. Qi, X. Zhang, Z. Hu, B. Xiang, R. Zhang, and S. Fang, “Choosing the right collaboration partner for innovation: a framework based on topic analysis and link prediction,” *Scientometrics*, vol. 127, no. 9, pp. 5519–5550, 2022.
- [8] L. Yang, X. Jiang, Y. Ji, H. Wang, A. Abraham, and H. Liu, “Gated graph convolutional network based on spatio-temporal semi-variogram for link prediction in dynamic complex network,” *Neurocomputing*, vol. 505, pp. 289–303, 2022.
- [9] S. Bates, T. Hastie, and R. Tibshirani, “Cross-Validation: What Does It Estimate and How Well Does It Do It?,” *J. Am. Stat. Assoc.*, pp. 1–12, doi: [10.1080/01621459.2023.2197686](https://doi.org/10.1080/01621459.2023.2197686).
- [10] M. Nie, D. Chen, and D. Wang, “Graph Embedding Method Based on Biased Walking for Link Prediction,” *Mathematics*, vol. 10, no. 20, p. 3778, 2022, doi: [10.3390/math10203778](https://doi.org/10.3390/math10203778).
- [11] S. Raschka, “Model evaluation, model selection, and algorithm selection in machine learning,” *arXiv:1811.12808*, 2018.
- [12] S. Noel and V. Swarup, “Dependency-Based Link Prediction for Learning Microsegmentation Policy,” in *Information and Communications Security: 24th International Conference, ICICS 2022*, Canterbury, UK, 2022, pp. 569–588.
- [13] G. Xu, X. Zhou, J. Peng, and C. Dong, “SCL-WTNS: A new link prediction algorithm based on strength of community link and weighted two-level neighborhood similarity,” *Int. J. Mod. Phys. B*, vol. 36, no. 20, p. 2250120, 2022.
- [14] F. Müller, “Link and edge weight prediction in air transport networks. An RNN approach,” *Physica A*, vol. 613, p. 128490, 2023.

- [15] A. Elsheikh, A.S. Ibrahim, and M.H. Ismail, "Sequence-to-sequence learning for link-scheduling in D2D communication networks," *J. Netw. Comput. Appl.*, vol. 212, p. 103567, 2023.
- [16] N.N. Daud, S.H.A. Hamid, C. Seri, M. Saadoon, and N.B. Anuar, "Scalable link prediction in Twitter using self-configured framework," *arXiv:2208.09798*, 2022.
- [17] Y. Xiu, K. Cao, X. Ren, B. Chen, and W.K.V. Chan, "Self-Similar Growth and Synergistic Link Prediction in Technology-Convergence Networks: The Case of Intelligent Transportation Systems," *Fractal Fract.*, vol. 7, no. 2, p. 109, 2023.
- [18] P. Sathre, A. Gondhalekar, and W.C. Feng, "Edge-Connected Jaccard Similarity for Graph Link Prediction on FPGA," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 2022, pp. 1–10.
- [19] W. Quan, M. Liu, N. Cheng, X. Zhang, D. Gao, and H. Zhang, "Cybertwin-driven DRL-based adaptive transmission scheduling for software defined vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4607–4619, 2022.
- [20] K.W. Cho, M. Cominelli, F. Gringoli, J. Widmer, and K. Jamieson, "Cross-Link Channel Prediction for Massive IoT Networks," *arXiv:2212.07663*, 2022.
- [21] C. Xing, Y. Li, C. Chen, F. Li, Z. Zeng, and X. Zou, "Determinantal point process-based new radio unlicensed link scheduling for multi-access edge computing," *World Wide Web*, vol. 25, no. 5, pp. 2215–2239, 2022.